**eNetwork Communications Server for Windows NT Ｐ－８ｇｓ 6.0 および eNetwork Ｑ－＝Ｊｋ・３＿ｅＫ１－７ｇｓ： Windows 95、Windows NT Ｐ－８ｇｓ J4.2**

IBM

# ７９Ｆ｀管理Ｗｍ０ｉ＿ｓ０

SC88-5631-01
英文原典：SC31-8480-01

**eNetwork Communications Server for Windows NT Ｐ－８ｇｓ 6.0 および eNetwork Ｑ－＝Ｊｋ・３＿ｅＫ１－７ｇｓ： Windows 95、Windows NT Ｐ－８ｇｓ J4.2**

# ７９Ｆ｀管理Ｗｍ０ｉ＿ｓ０

英文原典：SC31-8480-01
SC88-5631-01

ご注意

本書、および本書がサポートする製 J をご H 用になる前に、 693ページの『U 録B. 特記 v 項』にある情報をお読みください。

本書は、IBM eNetwork Communications Server for Windows NT バージョン 6.0 および eNetwork パーソナル・コミュニケーションズ Windows 95、Windows NT バージョン J4.2、および特に断りがない限りそれ以降のすべてのリリースに適用されます。


**原　典：** SC31-8480-01
eNetwork Communications Server
Version 6.0 for Windows** NT and
eNetwork Personal Communications
Version 4.2
for Windows 95 and Windows** NT
System Management Programming

**/　行：** 日本アイ・ビー・エム t 式会社

**担　当：** ナショナル・ランゲージ・サポート

第1刷 1998.7


この文書では、平成明朝体™W3、平成明朝体™W9、平成Qゴシック体™W3、平成Qゴシック体™W5、および平成Qゴシック体™W7をH用しています。この(書体*)は、(財）日本規 J 協会とH用契約を締結しH用しているものです。フォントとして無断複製することは禁 _ されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

# 目次

# 表

# 本書について

本書では、IBM eNetwork Communications Server for Windows NT および IBM eNetwork パーソナル・コミュニケーションズ for Windows 95、Windows NT をH用するプログラムを+発する方法について説明します。

IBM eNetwork Communications Server for Windows NT (本書では *Communications Server* という) は通信サービス (CS)・プラットフォームです。このプラットフォームは、ホスト・コンピューターおよび他のワークステーションと通信する Windows NT の}広いワークステーションのためのサービスを提供します。 Communications Server ユーザーは種々のリモート接続性オプションから選択できます。

IBM eNetwork パーソナル・コミュニケーションズ for Windows 95、Windows NT (本書では *パーソナル・コミュニケーションズ*という) は全機能のエミュレーターです。ホスト端末エミュレーションに加えて、以下の有用な機能が提供されます。

- ファイル転送
- ダイナミック構成
- Hいやすいグラフィカル・インターフェース
- SNA を基にしたクライアント・アプリケーション用 API
- TCP/IP を基にしたアプリケーションが SNA を基にしたネットワークで通信することを可能とする API

ほとんどのインスタンスで、パーソナル・コミュニケーションズおよび Communications Server 用プログラムの+発は、多数の同じ verb をサポートしている点では類wしていますが、いくらか相違点があります。 それは、アイコンのHい方に顕著に現れます。特定な点の詳細については、xページの『アイコン』を2照してください。本書では、「*プログラム*」はパーソナル・コミュニケーションズおよび Communications Server の両方をXして用いられます。パーソナル・コミュニケーションズのプログラムまたは Communications Server のプログラムだけに適用される場合は、特定のプログラム名が用いられます。

本書では、Windows** という語で Windows 95 と Windows NT** の両方をXします。また、ワークステーション という語は、サポートされているすべてのパーソナル・コンピューターをXして用いられます。パーソナル・コンピューターの特定のモデルやアーキテクチャーを明示している場合は、そのタイプに限定されます。

## 対象読者

本書は、ノード・オペレーター機能 (NOF) API メッセージをH用して、パーソナル・コミュニケーションズまたは Communications Server の操作の管理および照会を行ったり、 ASCII 構成ファイルをH用したりするプログラマーや+発者を対象にしています。

また、リモート（ホスト・フォーカル・ポイント）・ネットワーク管理アプリケーションとの通信に、パーソナル・コミュニケーションズおよび Communications Server

が提供する基礎的な管理サービス・サポートをH用するようなネットワーク管理アプリケーションを作成する+発者も対象読者に含まれます。

## 本書の使い方

本書は 2 t に分かれています。1ページの『第1 t パーソナル・コミュニケーションズおよび Communications Serverノード・オペレーター機能』には以下の章があります。

- 3ページの『第1章 はじめに』では、本書の目的を説明しています。
- 7ページの『第2章 本書で取り上げる verb の 5 要』では、ノード・オペレーター機能 API 構造体とその構造体がサポートしている verb について説明します。この章では、組み込まれている verb をカテゴリーごとに説明し、パーソナル・コミュニケーションズおよび Communications Server に用意されているその他の信号についても取り上げます。
- 17ページの『第3章 ノード・オペレーター機能のエントリー・ポイント』では、エントリー・ポイントのH張Rについて説明します。
- 第 4 章から第 12 章では、それぞれの verb の構文を説明します。 F verb の情報を含んだ構造体のコピーに基づいて、それぞれの入力項目の説明や戻りコードのリストを示しています。

655ページの『第2 t パーソナル・コミュニケーションズおよび Communications Server 管理サービス API』には以下の章があります。

- 657ページの『第13章 管理サービス API の紹介』では、管理サービス API について説明します。
- 661ページの『第14章 管理サービスのエントリー・ポイント』では、管理サービス verb のエントリー・ポイントについて説明します。
- 675ページの『第15章 管理サービス verb』では、それぞれの verb の構文を説明します。 F verb の情報を含んだ構造体のコピーに基づいて、それぞれの入力項目の説明や戻りコードのリストを示しています。

### " $ 3 s

本書では、特別な情報を示す場合に以下のアイコンをH用します。

## 本書の表記規則

パーソナル・コミュニケーションズまたは Communications Server ライブラリーでは、以下の = 記規則を H 用しています。本書ですべての = 記規則を H 用するとは限りません。

### Ｆ - ９ Ｈ規則

| | |
|---|---|
| **Bold** | Bold の文 z は、プログラムにおいて、またはコマンド・プロンプトで H 用できる verb、関数、パラメーターに H 用します。これらの値には大文 z 小文 z の区別があるので、本書に示されているとおりに入力する , 要があります。 |
| イタリック | イタリック体の文 z は、以下の場合に H 用します。<br>• プログラマーが値を X 定するための変数。<br>• リスト、チェック・ボックス、入力フィールド、押しボタン、メニュー選択項目などのウィンドウ制御の名前。本書では、ウィンドウに = 示されるとおりの名前を H 用しています。<br>• マニュアルの書名。<br>• 1 文 z は 1 文 z として、1 語は 1 語として H 用します。 c : *a* が出てきたら、これは決して *an* ではありません。 |
| ***Bold イタリック*** | Bold イタリック体は、語を強調するために H 用します。 |
| 大文 z | 大文 z は、プログラムにおいて、またはコマンド・プロンプトで H 用できる定数、ファイル名 、キーワード、オプションに H 用します。これらの値は、大文 z で入力しても小文 z で入力しても構いません。 |
| 二重引用 d | 二重引用 d は、ウィンドウ内に = 示されるメッセージに H 用します。一例は、エミュレーター・セッションの操作員情報域 (OIA) に = 示されるメッセージです。 |
| 例の書体 | 例の書体は、コマンド・プロンプトまたはウィンドウに入力する情報に H 用します。 |

### 数値規則

| | |
|---|---|
| 2 進数 | BX'xxxx xxxx' または BX'x' という形になります。ただし、場合によっては、テキストと一緒に = 示される形（『2 進数値 xxxx xxxx は...』）もあります。 |
| ビット位置 | 右端の 0（最下位ビット）から O まります。 |
| 10 進数 | 4 桁を越える 10 進数はメートル法で = 示されます。3 桁ごとの区切りには、コンマではなくスペースを H 用します。 たとえば、16147 という数値は、16 147 となります。 |
| 16 進数 | テキスト内では、16 進数 xxxx または単に X'xxxx' という形で H 用します。たとえば、「『隣接ノードのアドレスは、16 進数 5D であり、これは X'5d' として X 定します』」のようになります。 |

# 参考文献

詳しくは、 *5 説およびインストール*を 2 照してください。これには Communications Serverのライブラリーと関連 q 料の全説明が含まれています。

Communications Serverのインストール後に特定ブックをビューするには、読者のデスクトップで次のパスを H 用してください。

1. プログラム
2. IBM Communications Server
3. q 料
4. ブックのリストから選択

Communications Server ブックはポータブル文書形式 (PDF) であり、それは Adobe Acrobat Reader で = 示可能です。読者のマシンにこのプログラムをお } ちでない場合には、読者はそれを文書リストからインストールできます。

インターネットのCommunications Server ホーム・ページには APAR と修正についてサービス情報に加えて一般製 J 情報もあります。ホーム・ページを得るには、 IBM Web エクスプローラーなどのインターネット・ブラウザーを H 用して、次の URL に行きます。

**http://www.software.ibm.com/enetwork/commserver/about/csnt.html**

詳しくは、 *5 説およびインストール*を 2 照してください。これには Personal Communications のライブラリーと関連 q 料の全説明が含まれています。

パーソナル・コミュニケーションズのインストール後に特定ブックをビューするには、読者のデスクトップで次のパスを H 用してください。

1. プログラム
2. IBM Communications Server
3. q 料
4. ブックのリストから選択

Personal Communications ブックは BookManager 形式 (BOO) であり、それは IBM ライブラリー・リーダー (ILR) で = 示可能です。読者のマシンにこのプログラムをお } ちでない場合には、読者はそれを eNetwork Personal Communications CD-ROM からインストールできます。

インターネットの Personal Communications ホーム・ページには APAR と修正についてサービス情報に加えて一般製 J 情報もあります。ホーム・ページを得るには、IBM Web エクスプローラーなどのインターネット・ブラウザーを H 用して、次の URL に行きます。

**http://www.software.ibm.com/enetwork/pcomm/**

# 第1部 Ｑー＝Ｊｋ・３＿ｅＫ１ー７ｇｓ：および Communications Server Ｎー Ｉ・＊Ｚｌー？ー機能

# 第1章 はじめに

この t では、パーソナル・コミュニケーションズおよび Communications Server に用意されているノード・オペレーター機能 (NOF) API について説明します。

## 本書の目的

本書の目的は以下のとおりです。

- ノード・オペレーター機能 API の構造体について 5 要を説明すること。
- インターフェースを流れる信号の構文について説明すること。

## Ｑ ー = Ｊ ｋ ・ ３ ＿ ｅ Ｋ １ ー ７ ｇ ｓ ：および Communications Server Ｎ ー Ｉ ・ * Ｚ ｌ ー ？ ー機能

パーソナル・コミュニケーションズおよび Communications Server のノード・オペレーター機能によって、ノード・オペレーターと、制御点 (CP) および論理装置 (LU) との間で通信を行えます。ノード・オペレーター機能は、オペレーターからノード構成情報を受け取って、ノードの + O ~ に制御点を初期化します。 さらにこの機能は、ノード構成情報の照会要求や = 示要求も受け取ります。 ノード・オペレーターは以下のことを行えます。

- LU、DLC、ポート、リンクの定義と削除
- リンクとセッションの h 動化と非 h 動化
- 制御点と LU のデータベース情報や状況に関する情報の照会

ノード・オペレーターは、対話型ディスプレイを H 用する操作員、ファイル・インターフェースによってアクセスされるコマンド・ファイル、またはトランザクション・プログラムのいずれかになります。ノード・オペレーター機能は、verb インターフェースを H 用してノード・オペレーターと情報の伝達を行います。

## ( ｓ Ｈ ｊ ー ・ ] $ ｓ Ｈ

パーソナル・コミュニケーションズおよび Communications Server には、ノード・オペレーター機能の verb を処理するためのライブラリー・ファイルが用意されています。

ノード・オペレーター機能の verb には、わかりやすい言語インターフェースがあります。 *verb 制御ブロック* と呼ばれるメモリー・ブロック内のフィールドには、プログラマー作成のプログラムによって情報が書き込まれます。次に、読者のプログラムはエントリー・ポイントを呼び出して、verb 制御ブロックへポインターを渡します。この操作が終わると、ノード・オペレーター機能は、H 用した修正済みのフィールドを verb 制御ブロックに戻します。プログラムは、verb 制御ブロックからの戻りパラメーターを読み取ることができます。

ノード・オペレーター機能の verb のエントリー・ポイントは以下のとおりです。

- WinAsyncNOF()
- WinAsyncNOFEx()
- WinNOFCancelAsyncRequest()
- WinNOFCleanup()
- WinNOFStartup()
- WinNOFRegisterIndicationSink()
- WinNOFUnregisterIndicationSink()
- WinNOFGetIndication()

エントリー・ポイントの詳細については、17ページの『第3章 ノード・オペレーター機能のエントリー・ポイント』を 2 照してください。

## Verb 制御ＶｍＣ／ (VCB)

***プログラミングについての注:*** 基本オペレーティング・システムは、呼び出し側アプリケーションのアドレス空間でいくつかのサブシステムを実行することによって、パフォーマンスを最適化します。そのため、アプリケーション・プログラムによってローカル記述Ｒテーブル (LDT) の セレクターが誤用されると、操作に誤りが生じたり、システム障 2 が発生したりするおそれがあります。 したがって、アプリケーション・プログラムでは、ポインターの LDT セレクター・フィールドが変更されてしまうようなポインター；術演；を実行するべきではありません。

verb 制御ブロック (VCB) のためにＨ用するセグメントは、読み書き可能なデータ・セグメントである，要があります。プログラマー作成のプログラムでは、VCB をプログラム内の変数として宣言できます。あるいは、プログラム内に VCB をｄり当てたり、もっと大きなセグメントから VCB をｄり当てたりすることも可能です。 VCB は、プログラムが発行する verb のためのすべてのフィールドを収容できるだけの大きさにする，要があります。

verb が発行されてから処理が終了するまで、アプリケーション・プログラムでは、verb 制御ブロックの設定を変更するべきではありません。ノード・オペレーター機能は処理を終了すると、修正済みの完成した VCB を元のブロックにコピーして戻します。したがって、プログラムで verb 制御ブロックを変数として宣言する場合は、内ｔプロシージャーのスタック内よりも静的記憶装置内に宣言するほうがよいでしょう。

Ｆ VCB 内の予約済みの未Ｈ用フィールドにはすべてゼロ (X'00') を入れてください。実際のところ、verb 制御ブロック全体にゼロをＸ定してから、プログラムでパラメーターに値をｄり当てるほうが～間の節約になる場合があります。特に、予約済みフィールドにゼロを設定することは重要です。

**注:** VCB が読み書き可能ではない場合、または少なくとも 10 バイトの大きさ（ノード・オペレーター機能の 1 次戻りコードと 2 次戻りコードを収容できる大きさ）がない場合、ノード・オペレーター機能は VCB にアクセスできないため、基本オペレーティング・システムの処理が異常終了します。この終了は、*一般保護違反*、プロセッサー例 0 トラップ D として認識されます。

VCB が小さすぎる場合、または間違った種類のセグメントがH用されている場合は、ノード・オペレーター機能が INVALID_VERB_SEGMENT という 1 次戻りコードを戻します。

## Ｎ一Ｉ・＊Ｚｌ一？一機能 (NOF) のＷｍ０ｉ｀の作成

パーソナル・コミュニケーションズおよび Communications Server には、NOF の verb を処理するためのダイナミック・リンク・ライブラリー (DLL) ファイルが用意されています。

DLL は再入可能です。つまり、複数のアプリケーションのプロセスとスレッドが同~に DLL を呼び出せます。

NOF の verb には、わかりやすい言語インターフェースがあります。 *verb 制御ブロック* (VCB) と呼ばれるメモリー・ブロック内のフィールドには、プログラマー作成のプログラムによって情報が書き込まれます。そのプログラムは NOF の DLL を呼び出して、verb 制御ブロックへのポインターを渡します。この操作が終わると、NOF は、H用した修正済みのフィールドを VCB に戻します。プログラムは、verb 制御ブロックからの戻りパラメーターを読み取ることができます。

5ページの= 1 では、NOF プログラムのコンパイルとリンクに，要なシステム提供のヘッダー・ファイルとライブラリーのH用方法をソース・モジュールごとにまとめています。一 t のヘッダー・ファイルには、他の，要なヘッダー・ファイルが入っている場合があります。

*= 1. NOF のヘッダー・ファイルとライブラリー*

| オペレーティング・システム | ヘッダー・ファイル | ライブラリー | DLL 名 |
|---|---|---|---|
| WINNT & WIN95 | WINNOF.H | WINNOF32.LIB | WINNOF32.DLL |
| WIN3.1 | WINNOF.H | WINNOF.LIB | WINNOF.DLL |
| OS/2 | APPC_C.H | APPC.LIB | APPC.DLL |

## Communications Server SNA API ／ｉ＄＂ｓＨ・５］一Ｈ

この情報はCommunications Serverにのみ適用します。

Windows 95、Windows NT、Windows 3.1、および OS/2 オペレーティング・システム用のクライアントのセットが Communications Server に組み込まれています。これらのクライアントは、本書では SNA API クライアントと呼んでおり、全ノード・オペレーター機能のサブセットだけをサポートしています。たとえば、**WINNOF** は、Windows クライアント（95、NT、3.1）だけでサポートされている API です。サポートされている NOF の verb のリストを以下に示します。

- QUERY_LOCAL_LU
- QUERY_LU_0_TO_3
- QUERY_LU_POOL

- QUERY_MODE
- QUERY_MODE_DEFINITION
- QUERY_PARTNER_LU
- QUERY_PARTNER_LU_DEFINITION
- QUERY_PU
- QUERY_SESSION
- QUERY_TP
- QUERY_TP_DEFINITION

## Communications Server では５］ーＨされており、Ｑー＝Ｊｋ・３＿ｅＫ１ー７ｇｓ：では５］ーＨされていない verb

この情報はCommunications Serverにのみ適用します。

Communications Server ではサポートされており、パーソナル・コミュニケーションズではサポートされていない verb のリストを以下に示します。

- DEFINE_DOWNSTREAM_LU
- DEFINE_DOWNSTREAM_LU_RANGE
- DEFINE_DSPU_TEMPLATE
- DELETE_DOWNSTREAM_LU
- DELETE_DOWNSTREAM_LU_RANGE
- DELETE_DSPU_TEMPLATE
- QUERY_ADJACENT_NN
- QUERY_DIRECTORY_STATS
- QUERY_DOWNSTREAM_LU
- QUERY_DOWNSTREAM_PU
- QUERY_DSPU_TEMPLATE
- QUERY_HPR_STATS
- QUERY_ISR_SESSION
- QUERY_NN_TOPOLOGY_NODE
- QUERY_NN_TOPOLOGY_STATS
- QUERY_NN_TOPOLOGY_TG
- DOWNSTREAM_LU_INDICATION
- DOWNSTREAM_PU_INDICATION
- ISR_INDICATION
- NN_TOPOLOGY_NODE_INDICATION
- NN_TOPOLOGY_TG_INDICATION

# 第2章 本書で取り上げる verb の概要

本書で説明する verb インターフェースによって、プログラムはパーソナル・コミュニケーションズまたは Communications Server ネットワーク環境に関連Uけられている構成機能、システム管理機能、ノード定義機能の大 t 分を実行できます。この章では、それぞれの機能の 5 要を説明し、関連する verb について 5 説します。

## verb の説明の読み方

第 4 章から第 12 章では、構成、システム管理、接続マネージャーの verb について説明します。

### 指定Ｑｉａー？ー

それぞれの verb の説明には、 F パラメーターと、プログラムでX定するパラメーター値についての詳細な説明を載せたセクションがあります。

場合によっては、パラメーターに変数値をX定することが , 要です。

### 戻りＱｉａー？ー

それぞれの verb の説明には、 F パラメーターと、プログラムに戻されるパラメーター値についての詳細な説明を載せたセクションがあります。

#### 戻り３ーＩ

本書で説明する構成、システム管理、接続マネージャーの verb には、それぞれの戻りコードがあります。戻りコードは、verb の正常な実行またはエラー情報を示すものです。戻りコードは、 F verb の 『戻りパラメーター』 というセクションにリストされています。

#### 追加情報

多くの場合、verb の説明には 『追加情報』 というセクションがあります。このセクションには、verb についての役立つ追加情報が記載されます。

## 共通の VCB Ｕ＃ーｋＩ

この章では、ノード・オペレーター機能 API で渡されるF verb の構文を示します。また、 F verb にX定するパラメーターと戻されるパラメーターについて説明します。

```
typedef struct nof_hdr
{
        unsigned short  opcode;
        unsigned char   reserv2;         /* reserved */
        unsigned char   format;
        unsigned short  primary_rc;
        unsigned long   secondary_rc;
} NOF_HDR;
```

それぞれの VCB にはいくつかの共通フィールドがあります。それぞれの共通フィールドについて、次に説明します。

**opcode**
verb の演；コード。このフィールドで verb を識別します。

**format**
VCB の 形式を識別します。VCB の現行バージョンをX定するためにこのフィールドに設定する値については、それぞれの verb の個所で個別に説明します。

**primary_rc**
1 次戻りコード。 F verb の値については、それぞれの verb のセクションで取り上げます。

**secondary_rc**
2 次戻りコード。 1 次戻りコードが提供する情報を補足します。

## verb の要約

ノード・オペレーター機能 API でH用する様々な verb をHって、以下のことを行えます。

- ノード・リソースの構成
- リンクとセッションの h 動化と非 h 動化
- ノードが保 } している情報の照会
- セッション数の変更
- 非送信請求X示の処理
- パスワード・サポートの提供
- リモート LU に対する 『PING』 の実行
- CPI-C サイド情報の定義、照会、および削除

## Ｎ ー Ｉ 構成

リソースを定義するために、以下の verb をH用できます。

- DEFINE_ADJACENT_NODE
- DEFINE_CN
- DEFINE_COS
- DEFINE_DEFAULT_PU
- DEFINE_DLC
- DEFINE_DLUR_DEFAULTS
- DEFINE_DOWNSTREAM_LU

DEFINE_DOWNSTREAM_LU はCommunications Serverのみです。

- DELETE_PORT
- DELETE_TP

## 活動化と非活動化

リンク・レベルで以下の verb をH用できます。

- START_DLC
- START_LS
- START_PORT
- STOP_DLC
- STOP_LS
- STOP_PORT

以下の verb は、従属型 LU リクエスター機能のためにH用されます。

- START_INTERNAL_PU
- STOP_INTERNAL_PU

セッション・レベルで以下の verb をH用できます。

- ACTIVATE_SESSION
- DEACTIVATE_CONV_GROUP
- DEACTIVATE_SESSION

高性能経路X定 (HPR) の RTP 接続を強制実行してパスを切り換えるために、以下の verb をH用できます。

PATH_SWITCH

## Ｎ－Ｉの照会

以下の verb は、名前のUいたフィールドにノード情報を戻します。

- QUERY_DEFAULT_PU
- QUERY_DLUR_DEFAULTS
- QUERY_MDS_STATISTICS
- QUERY_NN_TOPOLOGY_STATS

QUERY_NN_TOPOLOGY_STATS はCommunications Serverのみです。

- QUERY_NODE
- QUERY_STATISTICS

以下の verb は、1 つまたは複数の情報を戻すことがあります。

- QUERY_ADJACENT_NN
- QUERY_ADJACENT_NODE
- QUERY_CN
- QUERY_CN_PORT
- QUERY_COS

- QUERY_DEFAULTS
- QUERY_DLUS
- QUERY_DOWNSTREAM_PU

QUERY_DOWNSTREAM_PU はCommunications Serverのみです。

- QUERY_DSPU_TEMPLATE
- QUERY_FOCAL_POINT
- QUERY_LU_POOL
- QUERY_LU62_TIMEOUT
- QUERY_MDS_APPLICATION
- QUERY_MODE_TO_COS_MAPPING
- QUERY_NMVT_APPLICATION
- QUERY_PU
- QUERY_TP

この情報は、リストの形で J 納されるものと見なせます。 verb では、リスト内に名前U きの記入項目をX定できるので、その名前U き記入項目がリスト内の位置マーカー（索引値）となります。これらの verb の **list_options** フィールドでは、リスト内のどの場所から情報を戻すかをX定します。

- **list_options** に AP_FIRST_IN_LIST という値を設定すると、索引値をX定したフィールドは無 k され、リストは先頭から戻されます。
- **list_options** に AP_LIST_INCLUSIVE という値を設定すると、リストはX定の索引値から戻されます。
- **list_options** に AP_LIST_FROM_NEXT という値を設定すると、リストはX定の索引値の後の記入項目から戻されます。

索引値では、戻り情報の + O点をX定します。 この値をX定すると、いくつかの照会 verb からも、戻りリストのためのフィルター処理オプションが得られます。 それらのフィルター処理オプションは、索引値とは独立した形でX定します。特にX定しなければ、戻りリストは IBM の 6611 APPN MIB に基づく順番になることに注意してください。 （verb パラメーターと MIB テーブル項目のマッピングについては、689ページの『U録A. IBM APPN MIB =』を 2 照してください。）

戻される項目の数または満杯のバッファー・サイズを設定します。 （両方を設定した場合は、verb によって、少ないほうのX定値に基づいて情報が戻されます。）通常、アプリケーションのバッファー・サイズによって、戻される情報量が制限されるので、ノード・オペレーター機能は、要求された情報を戻すために，要なバッファー・スペースの合計量と、そのスペースに収まる項目の合計数を示す追加情報を戻します。

以下の verb は、1 つまたは複数の情報を戻すだけではなく、 F 種レベルの情報を戻すことも可能です。 **list_options** フィールドでは、 AP_DETAIL か AP_SUMMARY のいずれかをX定することによって、詳細情報か要約のどちらを戻すかをX定します。 この 2 つのオプションは、上記の **list_options** で **いずれか**をX定します。たとえば、AP_DETAIL | AP_FIRST_IN_LIST のようになります。

- QUERY_DIRECTORY_LU

- QUERY_DLC
- QUERY_DLUR_LU
- QUERY_DLUR_PU
- QUERY_DOWNSTREAM_LU

QUERY_DOWNSTREAM_LU はCommunications Serverのみです。

- QUERY_ISR_SESSION

QUERY_ISR_SESSION はCommunications Serverのみです。

- QUERY_LOCAL_LU
- QUERY_LOCAL_TOPOLOGY
- QUERY_LS
- QUERY_LU_0_TO_3
- QUERY_MODE
- QUERY_MODE_DEFINITION
- QUERY_NN_TOPOLOGY_NODE

QUERY_NN_TOPOLOGY_NODE はCommunications Serverのみです。

- QUERY_NN_TOPOLOGY_TG

QUERY_NN_TOPOLOGY_TG はCommunications Serverのみです。

- QUERY_PARTNER_LU
- QUERY_PARTNER_LU_DEFINITION
- QUERY_PORT
- QUERY_RTP_CONNECTION
- QUERY_SESSION
- QUERY_TP_DEFINITION

## ;C7gs限度 verb

- CHANGE_SESSION_LIMIT
- INITIALIZE_SESSION_LIMIT
- RESET_SESSION_LIMIT

## 非送信請求指示

ノード情報を=示するアプリケーションでは、非送信請求X示（変更が生じたときに発行されて要約情報だけを戻すX示）をH用して、照会 verb（詳細情報を戻す）を呼び出すことができます。情報を受け取るためのアプリケーションが登録してあれ

ば、ノードは名前Uきのイベントによる非送信請求X示として、以下に挙げる信号を作

**WinNOFUnregisterIndicationSink**
登録抹消してX示を受け取らないようにする

**WinNOFGetIndication**
X示を受け取る

非送信請求X示は、ノード・オペレーター機能に登録されているX示シンクに渡されます。X示を生成した構成要素がそのX示を送信できない場合、その構成要素は、次回発行するX示に **data_lost** 8識を設定します。 X示の **data_lost** フラグがAP_YES とX定された場合は、それ以降のデータ・フィールドが NULL に設定されることがあります。このフラグは、変更が生じたものの、その変更の詳細情報が失われたことをアプリケーションに通知し、アプリケーションが適切な照会 verb を発行してそのv態に対応するべきことを知らせます。

信号 LULU_EVENT は、それがノードからの送信請求なしに verb REGISTER_LULU_EVENT と UNREGISTER_LULU_EVENT をH用して登録済みプロセスへ送信されるときに、X示としてクラス分けもされることに注意してください。それは上記にはリストされていません。その振るqいが次のように極めて異なるからです。登録は LU パートナー LU ペア用であり、 **data_lost** に同等のものはありません。これらの LULU イベントX示は , ず生成されます。

## 機密保護 verb

以下の機密保護 verb によって、 LU-LU 検査または会話機密保護のためのパスワードを管理できます。

- DEFINE_LU_LU_PASSWORD
- DEFINE_USERID_PASSWORD
- DELETE_LU_LU_PASSWORD
- DELETE_USERID_PASSWORD

## APING verb

以下の verb によって、管理アプリケーションからネットワーク内のリモート LU に対して 『ping』 を実行できます。

APING

## CPI-C の verb

以下の verb によって、 CPI-C サイド情報の定義、照会、削除を行えます。

- DEFINE_CPIC_SIDE_INFO
- DELETE_CPIC_SIDE_INFO
- QUERY_CPIC_SIDE_INFO

パーソナル・コミュニケーションズおよび Communications Server - Windows 95、Windows NT に用意されている CPI-C サポートに関する詳細については、*CPI-C Reference* を 2 照してください。

## 接続^Ｍ一８ｃ一 verb

接続マネージャーを制御するために、以下の verb をH用できます。

- DISABLE_ATTACH_MANAGER
- ENABLE_ATTACH_MANAGER
- QUERY_ATTACH_MANAGER

## DLC Ｗｍ；９、］一Ｈ、ｊｓ／・９Ｆ一７ｇｓ

### DLC Ｗｍ；９

パーソナル・コミュニケーションズまたは Communications Server は、複数の DLC プロセスを作成できます。それぞれの DLC プロセスは、ノード・オペレーター機能の API で発行される START_DLC verb への応答として、パーソナル・コミュニケーションズまたは Communications Server が作成するものです。F DLC は、特定のデータ・リンク・プロトコル（SDLC やトークンリングなど）をH用し、1 つのリンクまたは 1 セットのリンクを介して通信を行います。

1 つの DLC プロセスによって、1 つまたは複数のポートを管理できます。ポートについては、以下で説明します。

### ］一Ｈ

ポートとは、ローカル・マシン内の固有のアクセス・ポイント（MAC/SAP アドレス・ペアなど）であり、 1 つの DLC プロセスと関連Uけられます。1 つの DLC に 1 つまたは複数のポートを設定できます。ポートの種類は以下のとおりです。

**交 9 回線ポート**
: 1 つまたは複数の隣接リンク・ステーションを設定できます。複数の隣接リンク・ステーションを同~にアクティブすることが可能です。 （この点は *SNA APPN Architecture Reference* での定義と異なることに注意してください。）

**専用回線ポート**
: 2 地点間リンク接続と複数地点間リンク接続の両方を設定できます。専用回線リンク接続で結ばれる隣接リンク・ステーションは、ノード・オペレーター機能の構成要素によって定義する，要があります。複数地点間専用回線リンクの場合は、すべてのノードに 1 次関連と 2 次関連を正しく設定することによって、予期しない結果が生じないようにしてください。

**SATF ポート**
: トークンリングなどの共用アクセス転送機能をH用します。 このポートによって、共用アクセス転送機能に接続しているリンク・ステーション・ペア間の接続が可能になります。トークンリング上でアクティブにするすべてのリンク・ステーションの初期設定は、，ず変更可能として定義する，要があります。そうすれば、どのリンク・ステーションからでもリンクをアクティブにすることが可能になります。

  **注:** SATF ポートは、接続ネットワークに関連Uけることもできます。この場合、固有のアクセス・ポイントのアドレスをブロードキャストするために、トポロジーの更新 v 項がH用されます。

## ｊｓ／・９Ｆー７ｇｓ

リンク・ステーションはポートに関連Uけられるもので、隣接ノードへの接続を=します。ポートには、複数のリンク・ステーションを設定できます。リンク・ステーションは、以下のように分類できます。

**定A済みリンク・ステーション**
明示的に（DEFINE_LS verb をHって）定義されたリンク・ステーション。

**動的リンク・ステーション**
接続ネットワークを介して動的接続をアクティブにした結果として作成されるリンク・ステーション。仮想経路X定ノード (VRN) とも言います。

**暗黙リンク・ステーション**
交換回線ポートまたは SATF ポート上で以前は認識されていなかったパートナー・ノードから受け取った呼び出しの結果として作成されるリンク・ステーション。（このタイプのポートは *SNA APPN Architecture Reference* では定義されていません。）

**一時リンク・ステーション**
交換回線ポートまたは SATF ポート上の DLC インターフェースを介して CONNECT_IN を受け取った~に作成されるリンク・ステーション。このリンク・ステーションは、リモート・ノードが識別されると、削除されるか、動的または暗黙のリンク・ステーションになります。

# 第3章 Ｎ－Ｉ・＊Ｚｌ－？－機能の(ｓＨｊ－・］$ｓＨ

この章では、ノード・オペレーター機能の verb のエントリー・ポイントについて説明します。

# WinNOF()

この関数には、ノード・オペレーター機能のすべての verb の同期エントリー・ポイントが用意されています。

## 構文

```
void WINAPI WinNOF(long vcb,
                   unsigned short vcb_size)
```

| パラメーター | 説明 |
| --- | --- |
| **vcb** | verb 制御ブロックへのポインター。 |
| **vcb_size** | verb 制御ブロックのバイト数。 |

## 戻り値

戻り値はありません。 verb 制御ブロックの **primary_rc** および **secondary_rc** フィールドに値が入っている場合は、エラーが発生しています。

## 解説

これは、ノード・オペレーター機能 API の主要な同期エントリー・ポイントです。この呼び出しは verb の処理が終了するまでブロックされます。

# WinAsyncNOF()

この関数には、ノード・オペレーター機能のすべての verb の非同期エントリー・ポイントが用意されています。

## 構文

```
HANDLE WINAPI WinAsyncNOF(HWND hwnd,
                          long vcb,
                          unsigned short vcb_size)
```

| パラメーター | 説明 |
|---|---|
| **hwnd** | 完了メッセージを受け取るウィンドウ・ハンドル。 |
| **vcb** | verb 制御ブロックへのポインター。 |
| **vcb_size** | verb 制御ブロックのバイト数。 |

## 戻り値

戻り値によって、非同期要求が正常に処理されたかどうかを判別できます。関数が正常に処理された場合、実際の戻り値はハンドルになります。正常に処理されなかった場合は、ゼロが戻されます。

## 解説

このエントリー・ポイントのH用中には、それぞれのアプリケーション・スレッドには一度に 1 つの未処理要求しか存在できません。

非同期操作の終了~に、アプリケーションの「*hWnd*」ウィンドウには、入力文z列である **“WinAsyncNOF”** と一緒に **RegisterWindowMessage** というメッセージが戻されます。 *wParam* 引き数には、元の関数呼び出しによって戻された非同期タスク・ハンドルが入っています。

関数が正常に値を戻す場合、操作の終了~、または会話の取り消し~に、**WinAsyncNOF()** メッセージがアプリケーションに通知されます。

**注: WinNOFCancelAsyncRequest()** も2照してください。

# WinAsyncNOFEx()

この関数には、ノード・オペレーター機能のすべての verb の非同期エントリー・ポイントが用意されています。ブロック化呼び出しではなくこのエントリー・ポイントをH用して、複数の verb を同じスレッドで処理できるようにしてください。

## 構文

```
HANDLE WINAPI WinAsyncNOFEx(HANDLE handle,
                            long vcb,
                            unsigned short vcb_size);
```

| パラメーター | 説明 |
|---|---|
| **handle** | アプリケーションが待機するイベントのハンドル。 |
| **vcb** | verb 制御ブロックへのポインター。 |
| **vcb_size** | verb 制御ブロックのバイト数。 |

## 戻り値

戻り値によって、非同期要求が正常に処理されたかどうかを判別できます。 関数が正常に処理された場合、実際の戻り値はハンドルになります。

## 解説

このエントリー・ポイントは、Win32** API で WaitForMultipleObjects と一緒にH用するためのものです。この関数の詳細については、Win32 API のプログラミング文書を 2 照してください。

非同期操作が終了すると、終了イベントの信号によって、そのことがアプリケーションに通知されます。終了イベントの信号があった場合は、 1 次戻りコードと 2 次戻りコードを調べて、エラー状態がないかどうかをN認してください。

**注: WinNOFCancelAsyncRequest()** も 2 照してください。

# WinNOFCancelAsyncRequest()

この関数は、**WinAsyncNOF** による未処理要求を取り消します。

## 構文

```
int WINAPI WinNOFCancelAsyncRequest(HANDLE handle);
```

**パラメーター　説明**

**handle**　　X定パラメーター。取り消す要求のハンドルをX定します。

## 戻り値

戻り値によって、非同期要求が取り消されたかどうかを判別できます。 値がゼロの場合は、要求が取り消されました。そうでない場合は、以下の値が戻されます。

**WNOFALREADY**
取り消すべき非同期要求がすでに終了していたか、ハンドルが有効ではなかったことを示すエラー・コード。

## 解説

**WinAsyncNOF** のいずれかの関数によって発行された非同期要求を処理の終了前に取り消すには、最初の関数によって戻されたハンドルを *handle* にX定して、**WinNOFCancelAsyncRequest()** 呼び出しを発行します。

非同期要求を取り消すと、アプリケーションの verb 制御ブロックに対する更新が停 _ し、verb の終了を（ウィンドウ・メッセージかイベントのいずれかによって）通知されるアプリケーションが停 _ します。この処置では、基礎的な要求は取り消されません。基礎的な要求を実際に取り消すには、アプリケーションから適切な NOF verb（たとえば、 START_LS を取り消すには STOP_LS）を発行する , 要があります。

万一、**WinAsyncNOF** による既存の非同期ルーチンを取り消す n みが失敗し、WNOFALREADY というエラー・コードが出た場合、原因として考えられる v 態は 2 つあります。 1 つは、元のルーチンがすでに終了していて、その結果として出された通知をアプリケーションがすでに処理していた場合、もう 1 つは、元のルーチンはすでに終了していたものの、アプリケーションが終了通知をまだ処理していない場合です。

**注: WinAsyncNOF()** も 2 照してください。

# WinNOFCleanup()

この関数は、ノード・オペレーター機能の API からアプリケーションを終了または登録抹消します。

## 構文

```
BOOL WINAPI WinNOFCleanup(void);
```

## 戻り値

戻り値によって、登録抹消が正常に実行されたかどうかを判別できます。値がゼロでなければ、アプリケーションの登録抹消が正常に実行されています。ゼロの値が戻された場合は、アプリケーションの登録は抹消されていません。

## 解説

ノード・オペレーター機能の API からノード・オペレーター機能のアプリケーションを登録抹消する場合は、 **WinNOFCleanup()** をH用してください。

**WinNOFCleanup** をH用すると、 **WinNOFGetIndication** で待機しているスレッドが非ブロック化されます。これは、WNOFNOTREG（アプリケーションがX示を受け取るための登録をしていない）と一緒に戻されます。 **WinNOFCleanup** は、すべてのX示について、アプリケーションの登録を抹消します。 **WinNOFCleanup** は、同期であれ非同期であれ、すべての未処理 verb を AP_CANCELLED というエラーと一緒に戻します。ただし、その verb はノードの内 t で処理を終了します。

**WinNOFStartup** および **WinNOFCleanup** のH用は , 須ではありません。しかし、アプリケーションでは、この 2 つの呼び出しのH用が一貫している , 要があります。つまり、両方ともH用するか、両方ともH用しないかのどちらかにしてください。

**注: WinNOFStartup()** も 2 照してください。

# WinNOFStartup()

この関数によって、アプリケーションは , 要なノード・オペレーター機能 API のバージョンをX定し、製 J がサポートしている API のバージョンを検索できます。この関数は、アプリケーションが登録のためのノード・オペレーター機能 API の呼び出しを発行する前に呼び出すことができます。

## 構文

```
int WINAPI WinNOFStartup(WORD wVersionRequired,
                         LPWNOFDATA nofdata);
```

| パラメーター | 説明 |
|---|---|
| **wVersionRequired** | , 要なノード・オペレーター機能 API サポートのバージョンをX定します。高位バイトでマイナー・バージョン（改訂）をX定し、低位バイトでメジャー・バージョン番号をX定します。 |
| **nofdata** | ノード・オペレーター機能 API のバージョンと、API システムの説明を戻します。 |

## 戻り値

戻り値によって、アプリケーションが正常に登録されたかどうか、およびノード・オペレーター機能 API システムがX定のバージョン番号をサポートしているかどうかを判別できます。値がゼロの場合は、アプリケーションが正常に登録されました。また、X定のバージョンはサポートされています。それ以 0 の値は以下のいずれかになります。

**WNOFSYSERROR**
基礎となるネットワーク・サブシステムが、ネットワーク通信を行える状態になっていません。

**WNOFVERNOTSUPPORTED**
X定したノード・オペレーター機能 API サポートが、このシステムでは提供されていません。

**WNOFBADPOINTER**
nofdata パラメーターに誤りがあります。

## 解説

この呼び出しは、API の今後のリリースとの互換性をN保するためのものです。現行バージョンは 1.0 です。

**WinNOFStartup** および **WinNOFCleanup** のH用は , 須ではありません。しかし、アプリケーションでは、この 2 つの呼び出しのH用が一貫している , 要があります。つまり、両方ともH用するか、両方ともH用しないかのどちらかにしてください。

**注: WinNOFCleanup()** も 2 照してください。

# WinNOFRegisterIndicationSink()

この関数によって、アプリケーションは非送信請求X示を受け取るための登録を行えます。

## 構文

```
BOOL WINAPI WinNOFRegisterIndicationSink(unsigned short indication_opcode,
                                         unsigned short queue_size,
                                         unsigned short *primary_rc,
                                         unsigned long *secondary_rc);
```

| パラメーター | 説明 |
|---|---|
| **indication_opcode** | 登録して受け取るX示。 |
| **queue_size** | 受け取らないで待ち行列に入れるX示の数。ゼロをX定すると、現行値がH用されます（初期の省略~値は 10 に設定されています)。アプリケーションが登録するすべてのX示に対して、 1 つの待ち行列しかありません。 |
| **primary_rc** | 戻り値: 1 次戻りコード |
| **secondary_rc** | 戻り値: 2 次戻りコード |

## 戻り値

この関数の戻り値によって、登録が正常に実行されたかどうかを判別できます。値がゼロでなければ、登録が正常に実行されています。値がゼロの場合は、登録が正常に実行されませんでした。

## 解説

**WinNOFRegisterIndicationSink** をH用すれば、 **indication_opcode** でX定したタイプの非送信請求X示を受け取るための登録を行えます。

アプリケーションでは、受け取りたいとWっているX示のタイプごとに、**WinNOFRegisterIndicationSink** を発行する，要があります。

**注: WinNOFUnregisterIndicationSink** および **WinNOFGetIndication** も 2 照してください。

# WinNOFUnregisterIndicationSink()

この関数によって、アプリケーションは非送信請求X示の受け取りを中_します。

## 構文

```
BOOL WINAPI WinNOFUnregisterIndicationSink(unsigned short indication_opcode,
                                           unsigned short *primary_rc,
                                           unsigned long *secondary_rc);
```

| パラメーター | 説明 |
|---|---|
| **indication_opcode** | 登録を抹消して受け取りを中_するX示。 |
| **primary_rc** | 戻り値: 1 次戻りコード。 |
| **secondary_rc** | 戻り値: 2 次戻りコード。 |

## 戻り値

この関数の戻り値によって、登録抹消が正常に実行されたかどうかを判別できます。値がゼロでなければ、登録抹消が正常に実行されています。値がゼロの場合は、登録抹消が正常に実行されませんでした。

## 解説

**WinNOFUnregisterIndicationSink** をH用すれば、 **indication_opcode** でX定したタイプの非送信請求X示の受け取りを中_できます。

アプリケーションでは、受け取りを中_したいとWっているX示のタイプごとに、**WinNOFUnregisterIndicationSink** を発行する，要があります。

**注: WinNOFRegisterIndicationSink** および **WinNOFGetIndication** も 2 照してください。

# WinNOFGetIndication()

この関数によって、アプリケーションは非送信請求X示を受け取ります。

## 構文

```
int WINAPI WinNOFGetIndication(long buffer,
                               unsigned short *buffer_size,
                               unsigned long timeout);
```

| パラメーター | 説明 |
|---|---|
| **buffer** | X示を受け取るバッファーへのポインター。 |
| **buffer_size** | バッファーのサイズ。戻り値: X示のサイズ。 |
| **timeout** | X示を待機する~間（ミリC）。 |

## 戻り値

この関数の戻り値によって、X示が受け取られたかどうかを判別できます。

**0** X示が戻されました。

**WNOFTIMEOUT**
X示の待機~間が切れました。

**WNOFSYSNOTREADY**
基礎となるネットワーク・サブシステムが、ネットワーク通信を行える状態になっていません。

**WNOFNOTREG**
アプリケーションが、X示を受け取るための登録をしていません。

**WNOFBADSIZE**
バッファーが小さすぎるためにX示を受け取れません。十分な大きさがあるバッファーをX定して、 **WinNOFGetIndication** 呼び出しを再発行してください。X示のサイズは、**buffer_size** パラメーターで戻されます。

**WNOFBADPOINTER**
バッファーまたは **buffer_size** パラメーターのいずれかが無効です。

**WNOFSYSERROR**
予期しないシステム・エラーが起きました。

## 解説

これは、ブロック化呼び出しであり、以下のいずれかの状況で値を戻します。

- X示が戻された
- 待機~間が切れた
- アプリケーションが WinNOFCleanup 呼び出しを発行した
- 製J が停_した
- システム・エラーが起きた

**注: WinNOFRegisterIndicationSink** および **WinNOFUnregisterIndicationSink** も 2 照してください。

## WinNOFGetIndication()

# 第4章　Ｎ－Ｉ構成 verb

以下の verb をH用して、ノードの構成情報を定義したり削除します。

# DEFINE_ADJACENT_NODE

DEFINE_ADJACENT_NODE は、隣接ノード上の q 源用にノード・ディレクトリー・データベースに項目を追加します。

**注:** この verb は , 須ではなく、CP-CP セッションを H 用した隣接ノードへの h 動パスが存在する場合には発行してはなりません。

この verb は、エンド・ノード上で発行することができます。そのようにした場合、ノードの制御点がルート・ディレクトリーに追加されます。

ノードの制御点 LU を定義するには、以下のフィールドを設定します。

- **cp_name** フィールドにノードの制御点名を設定する
- ADJACENT_NODE_LU 構造体を追加し、**fqlu_name** フィールドに制御点名を X 定する

ノード上の追加の LU は、ノードの制御点の R としてディレクトリーに追加されます。 DEFINE_ADJACENT_NODE を H 用して、 LU 定義を既存のノード定義に追加することもできます。 LU は、DELETE_ADJACENT_NODE verb を発行するときと同じ方法で除去することができます。処理中に verb が失敗する場合、新しいディレクトリー項目はすべて除去され、ディレクトリーは verb を発行する前の状態になります。

## VCB 構造体

DEFINE_ADJACENT_NODE verb には、ADJACENT_NODE_LU オーバーレーの変数番号が含まれています。ADJACENT_NODE_LU 構造体は、DEFINE_ADJACENT_NODE 構造体の終わりに連結されます。

```
typedef struct define_adjacent_node
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   cp_name[17];         /* CP name                       */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   reserv3[19];         /* reserved                      */
        unsigned short  num_of_lus;          /* number of LUs                 */
} DEFINE_ADJACENT_NODE;

typedef struct adjacent_node_lu
{
        unsigned char   wildcard_lu;         /* wildcard LU name           */
                                             /* indicator                  */
        unsigned char   fqlu_name[17];       /* fully qualified LU name    */
        unsigned char   reserv1[6];          /* reserved                   */
} ADJACENT_NODE_LU;
```

## 指定 Q i a ー ? ー

アプリケーションは以下のパラメーターを X 定します。

**opcode**

AP_DEFINE_ADJACENT_NODE

**format**
VCB のフォーマットを識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**cp_name**
隣接エンド・ノードにおける制御点の完全修飾名。これは、ノードが XID（サポートされている場合）をH用して送信する名前、およびノードへのリンクのために DEFINE_LS 上でX定された隣接制御点名と一致していなければなりません。この名前の長さは 17 バイトであり、EBCDIC スペースが右に埋め込まれています。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

**description**
q 源の説明（QUERY_DIRECTORY_LU で戻される)。これは、ローカルに = 示可能な文 z セットによる 16 バイト（非ゼロ）のストリングです。16 バイトすべてに意味があります。

**num_of_lus**
DEFINE_ADJACENT_NODE VCB に続く隣接 LU オーバーレーの数。

**adjacent_node_lu.wildcard_lu**
X定した LU 名がワイルドカード名（AP_YES または AP_NO）であるかどうかを示します。

**adjacent_node_lu.fqlu_name**
定義する LU 名。この名前が完全修飾されていない場合、CP 名のネットワーク ID であると見なされます。この名前の長さは 17 バイトであり、EBCDIC スペースが右に埋め込まれています。 この名前は 1 つか 2 つのタイプ A の EBCDIC 文 z ストリングで構成され、これらは EBCDIC ドットで連結されています。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

**wildcard_lu** が TRUE のときは、EBCDIC スペースの後に続く ″.″ は 全機能のワイルドカード (何にでも一致する) です。すべての EBCDIC スペースは CP 名の Net ID でOまる何にでもマッチングします。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_CP_NAME

AP_INVALID_LU_NAME
AP_INVALID_WILDCARD_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_CP_NAME

AP_INVALID_LU_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

**secondary_rc**
AP_MEMORY_SHORTAGE

AP_DIRECTORY_FULL

# DEFINE_CN

DEFINE_CN は、接続ネットワーク（仮想経路ノードまたは VRN ともいう）を定義します。この verb により、接続ネットワークの伝送グループ (TG) 特性と共にネットワーク修飾名が提供されます。また、この接続ネットワークにアクセスするローカル・ポートの名前のリストも提供されます。

DEFINE_CN をH用して、既存の接続ネットワークを再定義できます。特に、もう一度 DEFINE_CN を発行することにより、接続ネットワークにアクセスするポートのリストに新しいポートを追加できます。(ポートは、DELETE_CN verb を発行するときと同じ方法で除去することができます。)

## VCB 構造体

```
typedef struct define_cn
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   attributes;          /* verb attributes                */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   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[RD_LEN];
                                             /* resource description           */
        unsigned char   num_ports;           /* number of ports on CN          */
        unsigned char   reserv1[16];         /* 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;         /* modem class                    */
        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;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_CN

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**fqcn_name**
定義する接続ネットワークの完全修飾名（17 バイト長)。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式でX定します。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

**def_data.description**
q 源の説明（QUERY_CN で戻される)。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**def_data.num_ports**
この接続ネットワークと関連したポート数。DEFINE_CN verb 1 つにつき 8 つのポートを設定でき、全体では CN につき 239 個以下のポートを設定できます。

**def_data.tg_chars.effect_cap**
有効な容量を示す実際の単位。この値は、1 バイトの b 動小数点数値としてコード化され、公式 0.1mmm * 2 eeeee で = されます。ここで、バイトのビット = 示は eeeeemmm です。有効な容量を示す F 単位は、1 C 当たり 300 ビットになります。

**def_data.tg_chars.connect_cost**
接続 ~ 間当たりのコスト。有効な値は、0 ～ 255 の範囲の整数値です。ここで、0 は接続 ~ 間当たりの最低コスト、255 は最高コストを示します。

**def_data.tg_chars.byte_cost**
バイト当たりのコスト。有効な値は、0～255 の範囲の整数値です。ここで、0 はバイト当たりのコストの最低を示し、255 は最高を示します。

**def_data.tg_chars.security**
下記のリストに示されている機密保護の値。

**AP_SEC_NONSECURE**
機密保護が存在しません。

**AP_SEC_PUBLIC_SWITCHED_NETWORK**
この接続ネットワークを介して伝送されるデータは、公衆交換ネットワークを介してフローします。

**AP_SEC_UNDERGROUND_CABLE**
保護地下ケーブルを介して伝送されたデータ。

**AP_SEC_SECURE_CONDUIT**
この伝送路は、保護されていない保護コンジットです。

**AP_SEC_GUARDED_CONDUIT**
コンジットは、物理的な漏れがないように保護されています。

**AP_SEC_ENCRYPTED**
伝送路を介した暗号化。

**AP_SEC_GUARDED_RADIATION**
伝送路は、物理的な漏れおよび電磁放射漏れがないように保護されています。

**def_data.tg_chars.prop_delay**
信号がリンクの長さを進むときにかかる ~ 間をマイクロ C で示す伝搬遅延。この値は、1 バイトの b 動小数点数値としてコード化され、公式 0.1mmm * 2 eeeee で = されます。ここで、バイトのビット = 示は eeeeemmm です。デフォルト値は、以下のとおりです。

**AP_PROP_DELAY_MINIMUM**
伝搬遅延なし。

**AP_PROP_DELAY_LAN**
480 マイクロ C 未満の遅延。

**AP_PROP_DELAY_TELEPHONE**
480 から 49 512 マイクロ C の遅延。

**AP_PROP_DELAY_PKT_SWITCHED_NET**
49 512 から 245 760 マイクロ C の間の遅延。

**AP_PROP_DELAY_SATELLITE**
245 760 マイクロ C より長い遅延。

**AP_PROP_DELAY_MAXIMUM**
伝搬遅延の最大値。

**def_data.tg_chars.modem_class**
予約済み。このフィールドは、常にゼロに設定されていなければなりません。

**def_data.tg_chars.user_def_parm_1**
0〜255 の範囲のユーザー定義パラメーター。

**def_data.tg_chars.user_def_parm_2**
0〜255 の範囲のユーザー定義パラメーター。

**def_data.tg_chars.user_def_parm_3**
0〜255 の範囲のユーザー定義パラメーター。

**port_name**
接続ネットワーク上で定義された 8 つまでのポート名の配列。名前のついたそれぞれのポートは、DEFINE_PORT verb によって定義されている , 要があります。 F ポート名は、ローカルに = 示可能な文 z による 8 バイトのストリングであり、関連する DEFINE_PORT verb での名前と一致していなければなりません。追加のポートは、もう一度 DEFINE_CN を発行して新しいポート名を X 定することにより、接続ネットワーク上で定義できます。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_CN_NAME

AP_INVALID_NUM_PORTS_SPECIFIED
AP_INVALID_PORT_NAME
AP_INVALID_PORT_TYPE
AP_DEF_LINK_INVALID_SECURITY
AP_EXCEEDS_MAX_ALLOWED

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PORT_ACTIVE

AP_CANT_MODIFY_VISIBILITY

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_COS

DEFINE_COS は、サービス・クラス定義を追加します。 DEFINE_COS verb をH用して、以前に定義した COS にあるフィールドを修正することもできます。

この定義により、ノードと TG「行」が提供されます。 これらの行は、ノードおよび TG 特性の範囲を、経路の計；にH用する重みに関連Uけます。重みが小さくなればなるほど、経路は順調になります。

## VCB 構造体

DEFINE_COS verb には、**cos_tg_row** および **cos_node_row** オーバーレーの変数番号が含まれています。**cos_tg_row** 構造体は、 DEFINE_COS の終わりに連結され（そして重みについて昇順に並べられる）、その後には **cos_node_row** 構造体が続きます（そして重みについて昇順に並べられる)。

```
typedef struct define_cos
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   cos_name[8];        /* class-of-service name         */
        unsigned char   description[RD_LEN]; /* resource description         */
        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_node_row
{
        COS_NODE_STATUS minimum;            /* minimum                       */
        COS_NODE_STATUS maximum;            /* max                           */
        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;

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;        /* modem class                  */
        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;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_COS

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**cos_name**

サービス・クラス名。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

**description**

ｑ源の説明（QUERY_COS で戻される)。これは、ローカルに＝示可能な文ｚセットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**transmission_priority**

伝送優先順位。これは、以下の値の 1 つに設定されます。

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

**num_of_node_rows**

DEFINE_COS VCB に続くノード行のオーバーレー数。最大値は 8 です。F ノード行には、一連のノード特性最小値、一連のノード特性最大値、そして重みが含まれます。特定のノードの重みを計；するときには、F ノード行に定義したノード特性最小値およびノード特性最大値に照らして、その特性が検査されます。次にこのノードには、最初のノード行の重みがｄり当てられます。これにより、すべてのノード特性をX定した限度内に制限されます。このノード特性に見合うノード行がリストされていない場合、このノードはこの COS にはT適当であると見なされ、無限の重みがｄり当てられます。ノード行は重みの昇順で連結する，要があることに注意してください。

**num_of_tg_rows**

ノード行のオーバーレーに続く TG 行のオーバーレー数。最大値は 8 です。F TG 行には、一連の TG 特性最小値、一連の TG 特性最大値、そして重みが含まれます。特定の TG の重みを計；するときには、F TG 行に定義した TG 特性最小値および TG 特性最大値に照らして、その特性が検査されます。次にこの TG には、最初の TG 行の重みがｄり当てられます。これにより、すべての TG の特性はX定した限度内に制限されます。この TG 特性に見合う TG 行がリストされていない場合、この TG はこの COS にはT適当

であると見なされ、無限の重みが d り当てられます。TG 行は重みの昇順で連結する , 要があることに注意してください。

**cos_node_row.minimum.rar**
経路追加抵抗最小値。値の範囲は、0〜255 でなければなりません。

**cos_node_row.minimum.status**
ノードの最小ふく輳状況を X 定します。値は以下のいずれかになります。

**AP_UNCONGESTED**
ノードがふく輳されていない。

**AP_CONGESTED**
ISR セッションの数は、 **isr_sessions_upper_threshold** の値を越えています。

**cos_node_row.maximum.rar**
経路追加抵抗最大値。値の範囲は、0〜255 でなければなりません。

**cos_node_row.maximum.status**
ノードの最大ふく輳状況を X 定します。値は以下のいずれかになります。

**AP_UNCONGESTED**
ノードがふく輳されていない。

**AP_CONGESTED**
ISR セッションの数は、 **isr_sessions_upper_threshold** の値を越えています。

**cos_node_row.weight**
このノード行と関連した重み。値の範囲は、0〜255 でなければなりません。

**cos_tg_row.minimum.effect_cap**
有効な容量を示す実際の単位についての最小限度。この値は、1 バイトの b 動小数点を H 用した数値としてコード化され、公式 0.1mmm * 2 eeeee で = されます。ここで、バイトのビット = 示は eeeeemmm です。有効な容量を示す F 単位は、1 C 当たり 300 ビットになります。

**cos_tg_row.minimum.connect_cost**
接続 ~ 間当たりのコストについての最小限度。有効な値は、0 〜 255 の範囲の整数値です。ここで、0 は接続 ~ 間当たりの最低コスト、255 は最高コストを示します。

**cos_tg_row.minimum.byte_cost**
バイト当たりのコストについての最小限度。有効な値は、0〜255 の範囲の整数値です。ここで、0 はバイト当たりのコストの最低を示し、255 は最高を示します。

**cos_tg_row.minimum.security**
下記のリストに示されている機密保護の値についての最小限度。

**AP_SEC_NONSECURE**
機密保護が存在しません。

**AP_SEC_PUBLIC_SWITCHED_NETWORK**
この接続ネットワークを介して伝送されるデータは、公衆交換ネットワークを介してフローします。

**AP_SEC_UNDERGROUND_CABLE**
保護地下ケーブルを介して伝送されたデータ。

**AP_SEC_SECURE_CONDUIT**
この伝送路は、保護されていない保護コンジットです。

**AP_SEC_GUARDED_CONDUIT**
コンジットは、物理的な漏れがないように保護されています。

**AP_SEC_ENCRYPTED**
伝送路を介した暗号化。

**AP_SEC_GUARDED_RADIATION**
伝送路は、物理的な漏れおよび電磁放射漏れがないように保護されています。

**cos_tg_row.minimum.prop_delay**
信号がリンクの長さを進むときにかかる ~ 間をマイクロ C で示す伝搬遅延についての最小限度。この値は、1 バイトの b 動小数点を H 用した数値としてコード化され、公式 0.1mmm * 2 eeeee で = されます。ここで、バイトのビット = 示は eeeeemmm です。デフォルト値は、以下のとおりです。

**AP_PROP_DELAY_MINIMUM**
伝搬遅延なし。

**AP_PROP_DELAY_LAN**
480 マイクロ C 未満の遅延。

**AP_PROP_DELAY_TELEPHONE**
480 から 49 512 マイクロ C の遅延。

**AP_PROP_DELAY_PKT_SWITCHED_NET**
49 512 から 245 760 マイクロ C の間の遅延。

**AP_PROP_DELAY_SATELLITE**
245 760 マイクロ C より長い遅延。

**AP_PROP_DELAY_MAXIMUM**
伝搬遅延の最大値。

**cos_tg_row.minimum.modem_class**
予約済み。このフィールドは、常にゼロに設定されていなければなりません。

**cos_tg_row.minimum.user_def_parm_1**
0〜255 の範囲のユーザー定義パラメーターについての最小限度。

**cos_tg_row.minimum.user_def_parm_2**
0〜255 の範囲のユーザー定義パラメーターについての最小限度。

**cos_tg_row.minimum.user_def_parm_3**
0〜255 の範囲のユーザー定義パラメーターについての最小限度。

**cos_tg_row.maximum.effect_cap**
有効な容量を示す実際の単位についての最大限度。この値は、1 バイトの b 動小数点を H 用した数値としてコード化され、公式 0.1mmm * 2 eeeee で = されます。ここで、バイトのビット = 示は eeeeemmm です。有効な容量を示す F 単位は、1 C 当たり 300 ビットになります。

**cos_tg_row.maximum.connect_cost**
接続~間当たりのコストについての最大限度。有効な値は、0 ～ 255 の範囲の整数値です。ここで、0 は接続~間当たりの最低コスト、255 は最高コストを示します。

**cos_tg_row.maximum.byte_cost**
バイト当たりのコストについての最大限度。有効な値は、0～255 の範囲の整数値です。ここで、0 はバイト当たりのコストの最低を示し、255 は最高を示します。

**cos_tg_row.maximum.security**
下記のリストに示されている機密保護の値についての最大限度。

**AP_SEC_NONSECURE**
機密保護が存在しません。

**AP_SEC_PUBLIC_SWITCHED_NETWORK**
この接続ネットワークを介して伝送されるデータは、公衆交換ネットワークを介してフローします。

**AP_SEC_UNDERGROUND_CABLE**
保護地下ケーブルを介して伝送されたデータ。

**AP_SEC_SECURE_CONDUIT**
この伝送路は、保護されていない保護コンジットです。

**AP_SEC_GUARDED_CONDUIT**
物理的な漏れがないように保護されているコンジット。

**AP_SEC_ENCRYPTED**
伝送路を介した暗号化。

**AP_SEC_GUARDED_RADIATION**
伝送路は、物理的な漏れおよび電磁放射漏れがないように保護されています。

**cos_tg_row.maximum.prop_delay**
信号がリンクの長さを進むときにかかる~間をマイクロCで示す伝搬遅延についての最大限度。この値は、1 バイトのb動小数点をH用した数値としてコード化され、公式 0.1mmm * 2 eeeee で=されます。ここで、バイトのビット=示は eeeeemmm です。デフォルト値は、以下のとおりです。

**AP_PROP_DELAY_MINIMUM**
伝搬遅延なし。

**AP_PROP_DELAY_LAN**
480 マイクロC未満の遅延。

**AP_PROP_DELAY_TELEPHONE**
480 から 49 512 マイクロCの遅延。

**AP_PROP_DELAY_PKT_SWITCHED_NET**
49 512 から 245 760 マイクロCの間の遅延。

**AP_PROP_DELAY_SATELLITE**
245 760 マイクロCより長い遅延。

**AP_PROP_DELAY_MAXIMUM**
伝搬遅延の最大値。

**cos_tg_row.maximum.modem_class**
予約済み。このフィールドは、常にゼロに設定されていなければなりません。

**cos_tg_row.maximum.user_def_parm_1**
0〜255 の範囲のユーザー定義パラメーターについての最大限度。

**cos_tg_row.maximum.user_def_parm_2**
0〜255 の範囲のユーザー定義パラメーターについての最大限度。

**cos_tg_row.maximum.user_def_parm_3**
0〜255 の範囲のユーザー定義パラメーターについての最大限度。

**cos_tg_row.weight**
この TG 行と関連した重み。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_COS_NAME

AP_INVALID_NUMBER_OF_NODE_ROWS
AP_INVALID_NUMBER_OF_TG_ROWS
AP_NODE_ROW_WGT_LESS_THAN_LAST
AP_TG_ROW_WGT_LESS_THAN_LAST

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_COS_TABLE_FULL

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_DEFAULTS

DEFINE_DEFAULTS をH用すると、ユーザーがノードのデフォルト・アクションを定義または再定義できるようになります。

## VCB 構造体

```
typedef struct define_defaults
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserve                       */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        DEFAULT_CHARS   default_chars;      /* default information           */
} DEFINE_DEFAULTS;

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

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_DEFAULTS

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**default_chars.description**

ｑ源の説明（QUERY_DEFAULTS で戻される)。これは、ローカルに＝示可能な文ｚセットによる 16 バイトのストリングです。 16 バイトすべてに意味があります。

**default_chars.mode_name**

デフォルト値として処理されるモードの名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

**default_chars.implicit_plu_forbidden**

「プログラム」 が未知のパートナー LU に対して暗黙の定義を正しく書き込んでいるかどうかを制御する。 (AP_YES または AP_NO)

**default_chars.specific_security_codes**

「プログラム」 が機密保護認証または許可障 2 に特定のセンス・コードをH

用するかどうかを制御する (AP_YES または AP_NO)。特定のセンス・コードは、セッションでそのコードのサポートをレポートしたパートナー LU にだけ戻されることに注意してください。

**default_chars.limited_timeout**
空きの限定 q 源競合勝者セッションが非 h 動化されるタイムアウトを X 定する。0 から 65535 C までの範囲

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

有効ではないデフォルト・モード（たとえば、EBCDIC A 以 0 ）、または有効だが定義されていないデフォルト・モードを verb が X 定する場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_MODE_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

F フィールドの再定義の結果は以下のとおりです。

**description**
再定義は即 ~ 有効になります。更新済み記述は次回の QUERY_DEFAULT verb で戻されます。

**mode_name**
再定義の結果はすべての後に続く暗黙モード定義に適用されます、たとえば、更新モードはデフォルト・モードとしサーブします。直前において未知のモード (たとえば、直前のデフォルト・モード文 z を継承したもの) への再定義の結果は未知のモードの再定義と同一です。たとえば、デフォルト・モ

ードが #INTER であり、「プログラム」 が (未知の) MODE1 用に bIND を受信する場合には、後で #BATCH に再定義されるデフォルト・モードの MODE1 への結果は、#BATCH のモード特性をX定する DEFINE_MODE (MODE1) の結果に同一です。

**implicit_plu_forbidden**
再定義は即~有効になります。後続のすべての暗黙 PLU 定義は、このフィールドの更新済み値によって、正常に行くか、または失敗するかです。

**specific_security_codes**
再定義は即~有効になります。

**limited_timeout**
更新済み値は再定義後にN立されたすべての新規作成セッションにH用されます。古い値は既存セッションにH用されます。

# DEFINE_DEFAULT_PU

DEFINE_DEFAULT_PU をH用すると、ユーザーはどのデフォルト PU フィールドについても定義、再定義、または修正ができるようになります。さらに、ヌルの PU 名をX定することにより、デフォルト PU を削除することもできます。 PU 名が TRANSFER_MS_DATA verb によって明示的にX定されないと、
この TRANSFER_MS_DATA によって伝送される管理サービス情報は、ホスト SSCP とのデフォルト PU のセッションによって送信されます。これについての詳細は、675 ページの『第15章 管理サービス verb』を 2 照してください。

## VCB 構造体

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

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_DEFAULT_PU

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**pu_name**

デフォルト値として処理されるローカル PU の名前。これは、8 バイト、英数z、タイプ A の EBCDIC 文z列（先頭は文z）です。EBCDIC スペースが右の余白に埋め込まれます。

**description**

q源の説明（QUERY_DEFAULT_PU で戻される)。これは、ローカルに=示可能な文zセットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

ノードがまだ+Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_DLC

DEFINE_DLC では、新しい DLC を定義するか、既存の DLC を修正します。この verb は、ノード内で固有な DLC 名と、基本構造体に連結されているいくつかの DLC 特有のデータを定義します。このデータは DLC の初期設定 ~ に H 用されますが、この形式はこの DLC タイプ（たとえばトークンリング）に特有のものです。 verb に追加された DLC 特有のデータに限り、DEFINE_DLC verb を H 用して修正できます。修正するためには、DLC がリセット状態になるように STOP_DLC verb を最初に発行しなければなりません。

DLC、ポート、およびリンク・ステーションの関連についての詳細は、15ページの『DLC プロセス、ポート、リンク・ステーション』を 2 照してください。

## VCB 構造体

```
typedef struct define_dlc
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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
{
        DESCRIPTION     description;         /* resource description          */
        unsigned char   dlc_type;            /* DLC type                      */
        unsigned char   neg_ls_supp;         /* negotiable LS support         */
        unsigned char   port_types;          /* allowable port types          */
        unsigned char   hpr_only;            /* DLC only supports HPR links:  */
        unsigned char   reserv3;             /* reserved                      */
        unsigned char   retry_flags;         /* conditions for automatic      */
                                             /* retries                       */
        unsigned short  max_activation_attempts;
                                             /* how many automatic retries?   */
        unsigned short  activation_delay_timer;
                                             /* delay between automatic       */
                                             /* retries                       */
        unsigned char    reserv4[4];         /* reserved                      */
        unsigned short  dlc_spec_data_len;   /* Length of DLC specific data   */
} DLC_DEF_DATA;
```

## 指定 Q i a ー ? ー

アプリケーションは以下のパラメーターを X 定します。

**opcode**

AP_DEFINE_DLC

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**dlc_name**

DLC の名前。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。OEM 装置の場合、この名前は製造元固有の名前になります。有効な値は、LAN、SDLC、AnyNet、X25 または TWINAX です（スペースUきの 8 文 z に埋め込まれる）。

**def_data.description**

q 源の説明（QUERY_DLC で戻される）。これは、ローカルに=示可能な文 z セットによる 16 バイトのストリングです。 16 バイトすべてに意味があります。

**def_data.dlc_type**

DLC のタイプ。パーソナル・コミュニケーションズおよび Communications Server は以下のタイプをサポートします。

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

**def_data.neg_ls_supp**

DLC が折衝可能リンク・ステーションをサポートするかどうかをX定します (AP_YES または AP_NO)。**dlc_type** が AP_TWINAX であれば、AP_NO だけがサポートされます。 **dlc_type** が AP_ANYNET であれば、AP_YES だけがサポートされます。

**def_data.port_types**

提供された **dlc_type** にH用可能なポート・タイプをX定します。この値は、以下の値の 1 つ以上が相互に OR 結合されたものに対応しています。

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

次の=をH用して、対応する DLC タイプのフィールドを設定します。

*= 2. DLC タイプのポート・タイプ*

| DLC タイプ | ポート・タイプ |
|---|---|
| AP_ANYNET | AP_PORT_SATF |
| AP_LLC2 | AP_PORT_SATF |
| AP_OEM_DLC | AP_PORT_SWITCHED or AP_PORT_NONSWITCHED |
| AP_SDLC | AP_PORT_SWITCHED or AP_PORT_NONSWITCHED |
| AP_TWINAX | AP_PORT_NONSWITCHED |
| AP_X25 | AP_PORT_SWITCHED or AP_PORT_NONSWITCHED |

**def_data.max_activation_attempts**

このフィールドは DLC が HPR リンクだけをサポートするかどうかをX定します。これは IP リンクの HPR に対して AP_YES を設定する，要があります。

AP_YES
AP_NO

**def_data.retry_flags**

このフィールドは、リンク・ステーションが自動的再 n 行の対象である条件をX定します。それはビット・フィールドで、以下の値のどれでもビット単位で互いに OR 結合したものが許されます。

**AP_RETRY_ON_START**

h 動化が n 行されたときにリモート・ノードから無応答の場合には、リンク h 動化が再 n 行されます。 h 動化が n 行されたときに基になっているポートが非 h 動である場合には、「プログラム」はそれを h 動化しようとします。

**AP_RETRY_ON_FAILURE**

リンクが h 動中、または h 動保留 ~ に障 2 があれば、リンク h 動化が再 n 行されます。 h 動化が n 行されたときに基になっているポートに障 2 があれば、「プログラム」はそれを h 動化しようとします。

**AP_RETRY_ON_DISCONNECT**

リンクがリモート・ノードにより正規に停 _ されれば、リンク h 動化が再 n 行されます。

**AP_DELAY_APPLICATION_RETRIES**

リンク h 動化は、アプリケーション (START_LS またはオンデマンド・リンク h 動化を H 用したもの) により + O されて、**activation_delay_timer** をH用してペースを取られます。

**AP_INHERIT_RETRY**

このフラグには効力がありません。

**def_data.max_activation_attempts**

このフィールドは、少なくとも 1 つのフラグが **def_data.retry_flags** の DEFINE_LS 内にセットされ、DEFINE_LS の **def_data.max_activation_attempts** が AP_USE_DEFAULTS にセットされ、かつ DEFINE_PORT の **def_data.max_activation_attempts** が AP_USE_DEFAULTS にセットされている、ということがなければ効力がありません。

このフィールドは、リモート・ノードが応答しないか、または基になっているポートが非 h 動であるときに、「プログラム」 が許可する再 n 行の数をX定します。これには自動的再 n 行とアプリケーション主導型の h 動化 n 行が含まれます。

この限&に達すると、それ以上の自動的再 n 行は行われません。この条件は STOP_LS、STOP_PORT、STOP_DLC または h 動化の成功によりリセットされ

ます。 START_LS または OPEN_LU_SSCP_SEC_RQ は 1 回 だけの h 動化 n 行に終わります、 h 動化が失敗しても再 n 行は行われません。

ゼロは '限&がない' ことを意味します。値 AP_USE_DEFAULTS は '限&がない' ことを意味します。

**def_data.activation_delay_timer**
このフィールドは、少なくとも 1 つのフラグが **def_data.retry_flags** の DEFINE_LS 内にセットされ、DEFINE_LS の **def_data.max_activation_attempts** が AP_USE_DEFAULTS にセットされ、かつ DEFINE_PORT の **def_data.max_activation_attempts** が AP_USE_DEFAULTS にセットされている、ということがなければ効力がありません。

このフィールドは、AP_DELAY_APPLICATION_RETRIES ビットが **def_data.retry_flags** にセットされている場合に、自動的再 n 行間に、およびアプリケーション主導型 h 動化 n 行間に 「プログラム」 が待つC数をX定します。

値、ゼロまたは AP_USE_DEFAULTS は 30 Cというデフォルトのタイマー~間のH用となります。

**def_data.dlc_spec_data_len**
このフィールドは、常にゼロに設定されていなければなりません。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DLC_NAME

AP_INVALID_DLC_TYPE
AP_INVALID_RETRY_FLAGS
AP_INVALID_PORT_TYPE
AP_HPR_NOT_SUPPORTED

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_DLC_ACTIVE

AP_INVALID_DLC_TYPE
AP_CANT_MODIFY_VISIBILITY

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_DLUR_DEFAULTS

DEFINE_DLUR_DEFAULTS をH用すると、ユーザーはデフォルトの従属 LU サーバー (DLUS) およびバックアップ・デフォルト DLUS を定義、再定義、または取り消しできるようになります。デフォルト DLUS 名は、関連した DLUS を明示的にX定していない PU 向けに SSCP-PU h動化を+ Oするときに、DLUR によってH用されます。 DLUS 名が DEFINE_DLUR_DEFAULTS verb によって明示的にX定されていない場合、現在のデフォルト（またはバックアップ DLUS）が取り消されます。

## VCB 構造体

```
typedef struct define_dlur_defaults
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   dlus_name[17];       /* DLUS name                     */
        unsigned char   bkup_dlus_name[17];  /* Backup DLUS name              */
        unsigned char   reserv3;             /* reserved                      */
        unsigned short  dlus_retry_timeout;  /* DLUS Retry Timeout            */
        unsigned short  dlus_retry_limit;    /* DLUS Retry Limit              */
        unsigned char   reserv4[16]          /* reserved                      */
} DEFINE_DLUR_DEFAULTS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_DLUR_DEFAULTS

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**description**

q源の説明。これは、ローカルに=示可能な文zセットによる 16 バイトのストリングです。 16 バイトすべてが有効なです。

**dlus_name**

デフォルトとして処理される DLUS ノードの名前。この名前は、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文zストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。) このフィールドがすべてゼロに設定されている場合、現在のデフォルト DLUS が取り消されます。

**bkup_dlus_name**

バックアップ・デフォルトとして処理される DLUS ノードの名前。この名前は、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文zストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。(それぞれの名

前は、埋め込みスペースがなく最大 8 バイトの長さです。）このフィールドがすべてゼロに設定されている場合、現在のバックアップ・デフォルト DLUS が取り消されます。

**dlus_retry_timeout**
DLUS への接続を 2 回以上 n みるときの間V（C数)。最初の n みと最初の再 n 行の間の間Vは、常に 1 Cです。ゼロをX定する場合、デフォルト値として 5 CがH用されます。

**dlus_retry_limit**
DLUS への接続が最初に失敗したとき以降に行われる再 n 行の最大数。ゼロをX定する場合、デフォルト値として 3 CがH用されます。 X'FFFF' をX定すると、パーソナル・コミュニケーションズまたは Communications Server は限度なしで再 n 行し続けます。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DLUS_NAME

AP_INVALID_BKUP_DLUS_NAME

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_DOWNSTREAM_LU

この verb はCommunications Serverにのみ適用します。

DEFINE_DOWNSTREAM_LU verb は、PU 集信用にH用されます。PU 集信をH用すると、ダウンストリームの LU はアップストリームのホストと通信できます。そのために、Communications ServerはF ダウンストリーム LU を従属のローカル LU にマップし、これはホスト *LU* として 2 照されるようになります。

DEFINE_DOWNSTREAM_LU は、新しいダウンストリーム LU を定義しますが、これをH用して既存の定義を修正することはできません。ダウンストリーム LU は、X定したホスト LU（DEFINE_LU_0_TO_3 verb をH用して定義済み）へマップされます。ホスト LU は、LU プールとしてもX定できます。

DEFINE_DOWNSTREAM_LU が既存のダウンストリーム LU 定義に発行されるときには、定義は等しくなければなりません。ダウンストリーム・リンクがアクティブで、かつダウンストリーム LU が非アクティブである場合には、 verb は成功として戻され、再 h 動化 n 行が行われます (しかし、これは成功しないこともありますが)。ダウンストリームがアクティブでないか、またはダウンストリーム LU がすでにアクティブである場合には、 verb は失敗しました。再 h 動化 n 行の処理はX定されたホスト LU の状態によります。

- ホスト LU がアクティブである場合には、ACTLU は即~にダウンストリーム LU に再送されます。
- ホスト LU が非アクティブである場合には、ノードはダウンストリーム LU に ACTLU を送信する前にホスト LU がアクティブとなるのを待ちます。ノードは、ホストへのリンクがアクティブでない場合には、それを h 動化することを n 行します (これは、ホスト・リンクが自動的に h 動化できない場合には成功しません)。

## VCB 構造体

```
typedef struct define_downstream_lu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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[RD_LEN]; /* resource description          */
        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;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_DOWNSTREAM_LU

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**dslu_name**

定義するダウンストリーム LU の名前。これは、8 バイト、英数z、タイプ A の EBCDIC 文z列（先頭は文z）です。EBCDIC スペースが右の余白に埋め込まれます。

**def_data.description**

q源の説明（QUERY_DOWNSTREAM_LU で戻される）。このフィールドの長さは、4 バイトの倍数でなければならず、ゼロであってはなりません。

**def_data.nau_address**

DOWNSTREAM LU のネットワーク・アドレス単位アドレス。この値の範囲は、1〜255 でなければなりません。

**def_data.dspu_name**

（DEFINE_LS でX定した）DOWNSTREAM PU の名前。これは、8 バイト、英数z、タイプ A の EBCDIC 文z列（先頭は文z）です。EBCDIC スペースが右の余白に埋め込まれます。

**def_data.host_lu_name**

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。これは、8 バイト、英数z、タイプ A の EBCDIC 文z列（先頭は文z）です。EBCDIC スペースが右の余白に埋め込まれます。

**def_data.allow_timeout**

セッションがホスト LU 定義 (AP_YES または AP_NO) でX定された **タイムアウト**期間に非アクティブのままになっている場合には、「プログラム」がこのダウンストリーム LU よってH用されるホスト LU をタイムアウトにすることを許可されるかどうかをX定します。

**def_data.delayed_logon**

最初のデータがダウンストリーム LU から受信されるまで、「プログラム」がホスト LU へのダウンストリーム LU の接続を遅らせるべきかどうかをX定します。その代わりとして、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 (AP_YES または AP_NO) に送信されます。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DNST_LU_NAME

AP_INVALID_NAU_ADDRESS

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE
AP_PU_NOT_DEFINED
AP_LU_ALREADY_DEFINED
AP_LU_NAU_ADDR_ALREADY_DEFD
AP_INVALID_HOST_LU_NAME
AP_LU_NAME_POOL_NAME_CLASH
AP_PU_NOT_ACTIVE
AP_LU_ALREADY_ACTIVATING
AP_LU_DEACTIVATING
AP_LU_ALREADY_ACTIVE
AP_CANT_MODIFY_VISIBILITY
AP_INVALID_ALLOW_TIMEOUT
AP_INVALID_DELAYED_LOGON
AP_DELAYED_VERB_PENDING

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_DOWNSTREAM_LU_RANGE

この verb はCommunications Serverにのみ適用します。

DEFINE_DOWNSTREAM_LU_RANGE verb は、PU 集信用にH用されます。PU 集信をH用すると、ダウンストリームの LU はアップストリームのホストと通信できます。そのために、Communications ServerはF ダウンストリーム LU を従属のローカル LU にマップし、これはホスト *LU* として2照されるようになります。

DEFINE_DOWNSTREAM_LU_RANGE をH用すると、X定した NAU 範囲の 1 つにおいて複数のダウンストリーム LU を定義できるようになります（ただし、これをH用して既存の定義を修正することはできない)。このノード・オペレーターが、ベース名と NAU 範囲をX定します。 LU 名は、ベース名と NAU アドレスを結合することにより生成されます。

たとえば、1 から 4 の NAU 範囲と結合された LUNME のベース名では、LU は LUNME001、LUNME002、LUNME003、および LUNME004 として定義されます。5 未満の非埋め込み文 z で構成されたベース名を定義すると、LU 名は 8 未満の非埋め込み文 z で構成されます。そうすると、Communications Serverは右に文 z を埋め込んで 8 文 z にします。

F ダウンストリーム LU は、X定したホスト LU（DEFINE_LU_0_TO_3 verb をH用して定義済み）へマップされます。

## VCB 構造体

```
typedef struct define_downstream_lu_range {
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   dslu_base_name[5];   /* Downstream LU base name       */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   min_nau;             /* min NAU address in range      */
        unsigned char   max_nau;             /* max 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 the    */
                                             /* host LU                       */
        unsigned char   reserv4[6];          /* reserved                      */
} DEFINE_DOWNSTREAM_LU_RANGE;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_DOWNSTREAM_LU_RANGE

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**dslu_base_name**

ダウンストリーム LU 名の範囲のベース名。これは、5 バイトのタイプ A の EBCDIC ストリング（文 z で + O）による英数 z であり、右側には EBCDIC スペースを埋め込みます。このベース名は、3 文 z のタイプ A の EBCDIC 数値文 z で構成されて 10 進数の NAU アドレスを示すものであり、これが NAU 範囲内の F LU に追加されます。

**description**

q 源の説明（QUERY_DOWNSTREAM_LU で戻される）。このフィールドの長さは、4 バイトの倍数でなければならず、ゼロであってはなりません。

**min_nau**

この範囲における NAU アドレスの最小値。この値は、包 g 的に 1 から 255 とすることができます。

**max_nau**

この範囲における NAU アドレスの最大値。この値は、包 g 的に 1 から 255 とすることができます。

**dspu_name**

（DEFINE_LS で X 定した）DOWNSTREAM PU の名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**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。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。 EBCDIC スペースが右の余白に埋め込まれます。

**allow_timeout**

セッションがホスト LU 定義 (AP_YES または AP_NO) で X 定された **タイムアウト**期間に非アクティブのままになっている場合には、「プログラム」がこのダウンストリーム LU よって H 用されるホスト LU をタイムアウトにすることを許可されるかどうかを X 定します。

**delayed_logon**

最初のデータがダウンストリーム LU から受信されるまで、「プログラム」がホスト LU へのダウンストリーム LU の接続を遅らせるべきかどうかを X 定します。その代わりとして、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 (AP_YES または AP_NO) に送信されます。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DNST_LU_NAME

AP_INVALID_NAU_ADDRESS
AP_INVALID_ALLOW_TIMEOUT
AP_INVALID_DELAYED_LOGON

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_LU_NAME_POOL_NAME_CLASH

AP_LU_ALREADY_DEFINED
AP_INVALID_HOST_LU_NAME
AP_PU_NOT_DEFINED
AP_INVALID_PU_NAME
AP_INVALID_PU_TYPE
AP_LU_NAU_ADDR_ALREADY_DEFD
AP_CANT_MODIFY_VISIBILITY
AP_DELAYED_VERB_PENDING

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_DSPU_TEMPLATE

この verb はCommunications Serverにのみ適用します。

この verb は、PU 集信用にH用されます。 PU 集信をH用すると、ダウンストリームの LU はアップストリームのホストと通信できます。そのために、Communications ServerはF ダウンストリーム LU を従属のローカル LU にマップし、これはホスト LU として2照されるようになります。 DEFINE_DSPU_TEMPLATE は、ダウンストリーム・ワークステーションのグループ上に存在するダウンストリーム LU のためのテンプレートを定義します。ワークステーションが暗黙のリンク（以前に定義されていないリンク）を介してCommunications Serverに接続するときに、このテンプレートをH用してダウンストリーム LU の定義を配置します。これらのテンプレートは、DEFINE_PORT verb の **implicit_dspu_template** フィールドによって2照されます。DEFINE_DSPU_TEMPLATE は、新しいテンプレートを定義するか、既存のテンプレートを修正するかのいずれかでH用できます（修正されたテンプレートの既存のインスタンスは影響を受けませんが)。

## VCB 構造体

```
typedef struct define_dspu_template
{
        unsigned short  opcode;               /* verb operation code          */
        unsigned char   attributes;           /* verb attributes              */
        unsigned char   format;               /* format                       */
        unsigned short  primary_rc;           /* primary return code          */
        unsigned long   secondary_rc;         /* secondary return code        */
        unsigned char   template_name[8];     /* name of template             */
        unsigned char   description;          /* resource description         */
        unsigned char   modify_template;      /* Modify existing template?    */
        unsigned char   reserv1[11];          /* reserved                     */
        unsigned short  max_instance;         /* Max active template          */
                                              /* instances                    */
        unsigned short  num_of_dslu_templates;
                                              /* number of DSLU templates     */
} DEFINE_DSPU_TEMPLATE;

typedef struct dslu_template
{
        unsigned char   min_nau;            /* min NAU address in range       */
        unsigned char   max_nau;            /* max 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;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_DSPU_TEMPLATE

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**template_name**

DSPU テンプレートの名前 （これは、PORT_DEF_DATA の **implicit_dspu_template** フィールドで X 定した名前に対応しています)。これは、ローカルに = 示可能な文 z セットによる 8 バイトのストリングです。8 バイトすべてが有効であり、すべて設定する , 要があります。

**description**

q 源の説明（QUERY_DSPU_TEMPLATE で戻される)。この長さは、4 バイトの倍数でなければならず、ゼロであってはなりません。

**modify_template**

この verb が追加の DSLU テンプレートを既存の DSPU テンプレートに追加すべきか、または既存の DSPU テンプレート (AP_MODIFY_DSPU_TEMPLATE または AP_REPLACE_DSPU_TEMPLATE) を置換すべきかどうかを X 定します。

変更テンプレートが AP_MODIFY_DSPU_TEMPLATE にセットされ、名前 U き DSPU テンプレートがない場合には、それが作成されます。

**modify_template** が AP_MODIFY_DSPU_TEMPLATE にセットされ、名前 U き DSPU テンプレートがない場合には、それが作成されます。

**modify_template** が AP_REPLACE_DSPU_TEMPLATE にセットされる場合には、新規作成テンプレートが作成されます。これは、包 g 的に 0 から 65535 とすることができます。0 は限度がないということです。

**max_instance**

これは、同 ~ に h 動状態にできるテンプレートのインスタンスの最大数です。この限度に達すると、新しいインスタンスを作成できません。これは、包 g 的に 0 から 65535 とすることができます。0 は限度がないということです。

**num_of_dslu_templates**

DEFINE_DSPU_TEMPLATE VCB に続く DSLU テンプレート・オーバーレーの数。この値は、包 g 的に 0 から 255 とすることができます。

**dslu_template.min_nau**

この範囲における NAU アドレスの最小値。この値は、包 g 的に 1 から 255 とすることができます。

**dslu_template.max_nau**

この範囲における NAU アドレスの最大値。この値は、包 g 的に 1 から 255 とすることができます。

**def_data.allow_timeout**
セッションがホスト LU 定義 (AP_YES or AP_NO) でX定された **タイムアウト**期間に非アクティブのままになっている場合には、「プログラム」 がこのダウンストリーム LU よってH用されるホスト LU をタイムアウトにすることを許可されるかどうかをX定します。

**def_data.delayed_logon**
最初のデータがダウンストリーム LU から受信されるまで、「プログラム」がホスト LU へのダウンストリーム LU の接続を遅らせるべきかどうかをX定します。その代わりとして、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 (AP_YES または AP_NO) に送信されます。

**dslu_template.host_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。これは、8 バイトのタイプ A の EBCDIC ストリング（文 z で + O）による英数 z であり、右側には EBCDIC スペースを埋め込みます。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TEMPLATE_NAME

AP_INVALID_NAU_ADDRESS
AP_INVALID_NAU_RANGE
AP_CLASHING_NAU_RANGE
AP_INVALID_NUM_DSPU_TEMPLATES
AP_INVALID_ALLOW_TIMEOUT
AP_INVALID_DELAYED_LOGON
AP_INVALID_MODIFY_TEMPLATE

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_HOST_LU_NAME

AP_CANT_MODIFY_VISIBILITY

関係のある START_NODE パラメーターがセットされなかったために verb が実行されない場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_FOCAL_POINT

パーソナル・コミュニケーションズまたは Communications Server には、異なったフォーカル・ポイントとの関連が幾種類もあります。DEFINE_FOCAL_POINT verb は、パーソナル・コミュニケーションズまたは Communications Server が暗黙に関連しているフォーカル・ポイントを定義します（このタイプは 1 次またはバックアップにできます)。この関連、およびそれらをN立する方法について下記で説明します。所定のカテゴリーの管理サービスのフォーカル・ポイント (FP) と管理サービスのエントリー・ポイント (EP) は、管理サービス機能メッセージを交換するときにN立されます。以下のタイプの FP-EP 関連がN立できます。

- 明示

  この関連は、エントリー・ポイントを制御の領域に d り当てるフォーカル・ポイントで、オペレーターによってN立されます。このフォーカル・ポイントは、管理サービス機能の交換を + O します。

- 暗黙（1 次）

  特定のエントリー・ポイントにいるオペレーターが、X定したフォーカル・ポイントにこのエントリー・ポイントを d り当てるときに、この関連がN立されます（たとえば、オペレーターが DEFINE_FOCAL_POINT verb を発行する場合)。この入り口点は、管理サービス機能の交換を + O します。

- 暗黙（バックアップ）

  エントリー・ポイントが明示の 1 次フォーカル・ポイントもしくは暗黙の 1 次フォーカル・ポイントのいずれかを消失する場合に、この関連がN立されます。このエントリー・ポイントは、管理サービス機能の交換を + O します。バックアップ・フォーカル・ポイントの ID は、（DEFINE_FOCAL_POINT verb をH用して）定義するか、管理サービス機能の交換によって取得することができます。

- デフォルト

  オペレーターが介入せずに FP が EP を取得する場合に、この関連がN立されます。FP は、MS 機能 の交換を + O します。この関連は NN である EP にのみ当てはまります。

- ドメイン

  ：当するネットワーク・ノード (NN) が、エンド・ノードのエントリー・ポイントにフォーカル・ポイントの ID を通知するときに、この関連がN立されます。ドメインの関連は、エンド・ノードにおいてだけ有効です。

- ホスト

  この関連では、管理サービス機能の交換は関係がなく、エントリー・ポイントノードからホストまでの SSCP-PU セッションの構成によってN立されます。これは、フォーカル・ポイントの関連では優先順位が一番低くなります。

F DEFINE_FOCAL_POINT verb だけをH用して、暗黙のフォーカル・ポイント（1 次タイプまたはバックアップ・タイプのいずれかが可）を定義できます。 F DEFINE_FOCAL_POINT verb は、特定の管理サービス・カテゴリー向けに発行されます。このカテゴリー内では、DEFINE_FOCAL_POINT verb をH用して以下を行えます。

- フォーカル・ポイントの定義

- フォーカル・ポイント（またはバックアップ・フォーカル・ポイント）の置き換え
- 現在 h 動化されているフォーカル・ポイントの取り消し

DEFINE_FOCAL_POINT verb のフィールドは、以下のようにしてH用します。

**ms_category** には、常に値が入っていなければなりません。**fp_fqcp_name** と **ms_appl_name** フィールドを組み合わせることにより、X定したカテゴリーのフォーカル・ポイント（**backup** フィールドが AP_YES に設定されている場合はバックアップ・フォーカル・ポイント）をX定します。

新しいフォーカル・ポイントを作成せずに、verb を発行して現在 h 動化されているフォーカル・ポイントを取り消す場合、**fp_fqcp_name** と **ms_appl_name** フィールドはすべてゼロに設定しなければなりません。フォーカル・ポイントを定義している（または置き換えている）DEFINE_FOCAL_POINT verb を受け取ると、パーソナル・コミュニケーションズまたは Communications Server は管理サービス機能要求を送信することにより、X定したフォーカル・ポイントと暗黙の 1 次フォーカル・ポイントとの関連をN立しようとします。現在のフォーカル・ポイントを取り消す DEFINE_FOCAL_POINT verb をパーソナル・コミュニケーションズまたは Communications Server が受け取ると、そのフォーカル・ポイントに管理サービス機能取り消しメッセージを送信します。DELETE_FOCAL_POINT verb（AP_ACTIVE をX定）をH用して、現在 h 動化されているフォーカル・ポイントを取り消すようお勧めします。

## VCB 構造体

```
typedef struct define_focal_point
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   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[RD_LEN];
                                             /* resource description           */
        unsigned char   backup;              /* is focal point a backup        */
        unsigned char   reserv3[16];         /* reserved                       */
} DEFINE_FOCAL_POINT;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_FOCAL_POINT

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**ms_category**

管理サービスのカテゴリー。これは、SNA 管理サービスで説明されている管

理サービス・カテゴリーの 4 バイトの体系定義値（右側に EBCDIC スペースが埋め込まれている）の 1 つとするか、8 バイトのタイプ 1134 EBCDIC 導入定義名のいずれかにできます。

**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、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文 z ストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。）このフォーカル・ポイントを取り消す場合、このフィールドはすべてゼロに設定しなければなりません。

**ms_appl_name**
フォーカル・ポイントのアプリケーション名。これは、「SNA Management Services」で説明されている管理サービス・アプリケーションの 4 バイトの体系定義値 (右側に EBCDIC スペースが埋め込まれている）の 1 つとするか、8 バイトのタイプ 1134 EBCDIC 導入定義名のいずれかにできます。このフォーカル・ポイントを取り消す場合、このフィールドはすべてゼロに設定しなければなりません。

**description**
q 源の説明（QUERY_FOCAL_POINT で戻される)。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**backup**
バックアップ・フォーカル・ポイントを定義するかどうかを X 定します（AP_YES または AP_NO)。現在 h 動化されているフォーカル・ポイントを取り消す場合、このフィールドは予約済みとなります。
DELETE_FOCAL_POINT verb（AP_ACTIVE を X 定）を H 用して、現在 h 動化されているフォーカル・ポイントを取り消すようお勧めします。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_FP_NAME

AP_INVALID_CATEGORY_NAME

verb が正常に実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**

AP_REPLACED

AP_UNSUCCESSFUL

**secondary_rc**

AP_IMPLICIT_REQUEST_REJECTED

AP_IMPLICIT_REQUEST_FAILED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_STOPPING

システム・エラーの場合、または「プログラム」が正常にフォーカル・ポイントに接続できなかった場合、「プログラム」は次のパラメーターが戻されます。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_INTERNAL_PU

DEFINE_INTERNAL_PU verb は、 DLUR 対応のローカル PU を定義します。この verb では、ホストに直接接続されるローカル PU は定義しません。それについては、81ページの『DEFINE_LS』を 2 照してください。

**注:** DEFINE_LS verb は以下を定義するためにH用されるべきです。

- 下記によりサーブされるダウンストリーム PU
  - DLUR
  - PU 集信
- ホストに直接接続されるローカル側ホスト

## VCB 構造体

```
typedef struct define_internal_pu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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[RD_LEN]; /* resource description          */
        unsigned char   dlus_name[17];       /* DLUS name                     */
        unsigned char   bkup_dlus_name[17];  /* backup DLUS name              */
        unsigned char   pu_id[4];            /* PU identifier                 */
        unsigned short  dlus_retry_timeout;  /* DLUS retry timeout            */
        unsigned short  dlus_retry_limit;    /* DLUS retry limit              */
        unsigned char   conventional_lu_compression;
                                             /* Data compression              */
                                             /* requested for con-            */
                                             /* ventional LU sessions         */
        unsigned char   conventional_lu_cryptography;
                                             /* Cryptography required         */
                                             /* for conventional LU           */
                                             /* sessions                      */
        unsigned char   reserv2[2];          /* reserved                      */
} INTERNAL_PU_DEF_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_INTERNAL_PU

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**pu_name**

定義する内t PU の名前。これは、8 バイト、英数z、タイプ A の EBCDIC 文z列（先頭は文z）です。EBCDIC スペースが右の余白に埋め込まれます。

**def_data.description**

q源の説明（QUERY_DLUR_PU および QUERY_PU で戻される）。これは、ローカルに=示可能な文zセットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**def_data.dlus_name**

SSCP-PU のh動化を+Oするときに DLUR がH用する DLUS ノードの名前。この名前は、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文zストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。（それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。）このフィールドがすべてゼロに設定されている場合、グローバルなデフォルト DLUS（DEFINE_DLUR_DEFAULTS verb をH用して定義されている場合）は DLUR で+Oした SSCP-PU h動化を行うときにH用されます。

**def_data.bkup_dlus_name**

この PU のバックアップ DLUS としてH用する DLUS ノードの名前。この名前は、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文zストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。（それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。）このフィールドがすべてゼロに設定されている場合、グローバルなバックアップ・デフォルト DLUS（DEFINE_DLUR_DEFAULTS verb をH用して定義されている場合）は、この PU のバックアップとしてH用されます。

**def_data.pu_id**

PU の ID。これは、4 バイトの16 進数文z列です。0〜11 のビットは、ブロック番号に設定されており、12〜31 のビットは、この PU を固有に識別する ID 番号に設定されています。これは、ホストで構成された **pu_id** と一致していなければなりません。

**def_data.dlus_retry_timeout**

**def_data.dlus_name** と **def_data.bkup_dlus_name** フィールドにX定された DLUS への接続を 2 回以上nみるときの間V（C数）。最初のnみと最初の再n行の間の間Vは、常に 1 Cです。ゼロをX定すると、DEFINE_DLUR_DEFAULTS を介して構成されたデフォルト値がH用されます。**def_data.dspu_services** が AP_DLUR に設定されていなければ、このフィールドは無kされます。

**def_data.dlus_retry_limit**

**def_data.dlus_name** と **def_data.bkup_dlus_name** フィールドにX定された DLUS への接続が最初に失敗した後で再n行する最大の回数。ゼロをX定すると、DEFINE_DLUR_DEFAULTS を介して構成されたデフォルト値がH用

されます。X'FFFF' がX定されると、「プログラム」は無限に再 n 行します。**def_data.dspu_services** が AP_DLUR に設定されていなければ、このフィールドは無 k されます。

**def_data.conventional_lu_compression**
データ圧縮をこの PU に依存する在来の LU セッションに要求するかどうかをX定します。

**AP_NO**
ローカル・ノードは、この PU をH用する在来の LU セッションでのデータ・フローを圧縮も圧縮解除もすべきではありません。

**AP_YES**
データ圧縮は、ホストが圧縮を要求している場合には、この PU に依存する LU セッションにH用可能化されるべきです。この値がセットされているが、ノードが圧縮 (START_NODE verb で定義された) をサポートしない場合には、INTERNAL_PU は正常に定義されますが、圧縮サポートはありません。

**def_data.conventional_lu_cryptography**
セッション・レベルの暗号化がこの PU に依存する在来の LU セッションに，要であるかどうかをX定します。

**AP_NONE**
ローカル・ノードは、この PU をH用する在来の LU セッションでのデータ・フローを圧縮も圧縮解除もすべきではありません。

**AP_MANDATORY**
，須のセッション・レベル暗号化は、インポート・キーが LU に選択可能であれば、APPN により実行されます。そうでない場合には、暗号化は、LU をH用するアプリケーションにより実行される，要があります(これが PU 集信である場合には、ダウンストリーム LU により行われます)。

**AP_OPTIONAL**
この値は暗号化がセッションごとにホスト・アプリケーションによる駆動にH用されることを許可します。セッションに対するホスト要求の暗号化がこの PU に依存する場合には、「プログラム」 の振る q いは AP_MANDATORY の場合と同じです。ホストが暗号化を要求しない場合には、振る q いは AP_NONE と同じです。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_PU_NAME

AP_INVALID_PU_ID
AP_INVALID_DLUS_NAME
AP_INVALID_BKUP_DLUS_NAME
AP_INVALID_CLU_CRYPTOGRAPHY

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_STATE_CHECK

**secondary_rc**

AP_PU_ALREADY_DEFINED

AP_CANT_MODIFY_VISIBILITY

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_LOCAL_LU

DEFINE_LOCAL_LU verb は、X定した特性のあるローカル LU を定義するよう要求します。または、LU がすでに存在している場合、LU の **attach_routing_data** 特性の修正を要求します。DEFINE_LOCAL_LU をH用して既存の定義を修正する場合、**attach_routing_data** フィールド以 0 のパラメーターはすべて無 k されます。

## VCB 構造体

### Format 1

```
typedef struct define_local_lu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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;         /* resource description          */
        unsigned char   lu_alias[8];         /* local LU alias                */
        unsigned char   nau_address;         /* NAU address                   */
        unsigned char   syncpt_support;      /* is sync-point supported?      */
        unsigned short  lu_session_limit;    /* LU session limit              */
        unsigned char   default_pool;        /* member of default_lu_pool     */
        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    */
        unsigned char   attach_routing_data; /* routing data for              */
                                             /* incoming attaches             */
        unsigned char   lu_model;            /* LU model for SDDLU            */
        unsigned char   model_name[7];       /* LU model name                 */
                                             /* for SDDLU                     */
        unsigned char   reserv4[16];         /* reserved                      */
} LOCAL_LU_DEF_DATA;
```

## VCB 構造体

### Format 0

```
typedef struct define_local_lu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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;         /* resource description          */
        unsigned char   lu_alias[8];         /* local LU alias                */
        unsigned char   nau_address;         /* NAU address                   */
```

```
        unsigned char   syncpt_support;      /* is sync-point supported?       */
        unsigned short  lu_session_limit;    /* LU session limit               */
        unsigned char   default_pool;        /* member of default_lu_pool      */
        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     */
        unsigned char   attach_routing_data; /* routing data for               */
                                             /* incoming attaches              */
} LOCAL_LU_DEF_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_LOCAL_LU

**format**

VCB の 形式を識別します。上記にリストされた VCB の format 0 または format 1 の一方をX定するには、このフィールドをゼロまたは 1 に設定します。

**lu_name**

定義するローカル LU の名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

**def_data.description**

ｑ源の説明（QUERY_LOCAL_LU で戻される)。これは、ローカルに＝示可能な文ｚセットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**def_data.lu_alias**

定義するローカル LU のエイリアス。これは、ローカル＝示可能文ｚセットの 8 バイトの文ｚ列です。8 バイトすべてが有効であり、すべて設定する，要があります。

**def_data.nau_address**

LU のネットワーク・アドレス単位アドレス。この値の範囲は 0〜255 でなければなりません。ゼロ以0の値であれば、この LU が従属 LU であることを示します。ゼロは、LU が独立 LU であることを暗黙X定します。

**def_data.syncpt_support**

同期点管理プログラムがこの LU ではH用できない場合は、このフィールドを常に AP_NO に設定してください。

**def_data.lu_session_limit**

LU によってサポートされているセッションの最大数。ゼロの値は限度がないことを示します。LU が独立している場合、この値は任意の値に設定できます。LU が従属であれば、この値を 1 に設定しなければなりません。

**def_data.default_pool**

LU が従属型 LU6.2 の省略～値プールのメンバーである場合には、AP_YES に設定します。

**def_data.pu_name**

この LU がH用する PU の名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。このフィールドは従属 LU によってのみH用され、独立 LU の場合はこのフィールドをすべて 2 進ゼロに設定しなければなりません。

**def_data.lu_attributes**

LU についての詳細をX定します。このフィールドは値 AP_NONE か、または 1 つ以上の下記オプションの互いに OR 結合されたものを取ります。

**AP_DISABLE_PWSUB**

ローカル LU 用パスワード置換サポートをH用T可にします。

**def_data.sscp_id**

これは、この LU をh動化するために、許可された SSCP の ID をX定します。それは 6 バイトの 2 進数フィールドです。このフィールドは、従属型 LU によってのみH用され、独立型 LU の場合、または LU が SSCP によりh動化された場合には、すべて 2 進ゼロに設定されます。

**def_data.disable**

LOCAL LU がH用T可とH用可能のどちらにされるかをX示します。LU は、このパラメーターを正しく (AP_YES または AP_NO) セットして DEFINE_LOCAL_LU を再発行することにより動的にH用可能またはH用T可にできます。H用T可の LU がH用可能となるときには、「プログラム」は NOTIFY (オンライン) を発行します。H用可能の LU がH用T可となるときには、「プログラム」は NOTIFY (オフライン) を発行します。LU がH用T可になるときにそれがバインド済みでれば、「プログラム」は UNBIND を発行し、それに NOTIFY (オフライン) が続きます。

**def_data.attach_routing_data**

接続経路X定データのタイプ。

**AP_REGISTERED_OR_DEFAULT_ATTACH_MGR**

このローカル LU でのトランザクション・プログラム (TP) への接続による DYNAMIC_LOAD_INDICATION は、この LU の DLI を受け取るよう登録されている接続マネージャーに送信されるか、この LU 向けに登録された接続マネージャーがなければデフォルトの接続マネージャーに送信されます。

**AP_REGISTERED_ATTACH_MGR_ONLY**

このローカル LU でのトランザクション・プログラム (TP) への接続による DYNAMIC_LOAD_INDICATION は、この LU の DLI を受け取るよう登録されている接続マネージャーだけに送信されます。この LU 向けに登録された接続マネージャーがなければ、接続は拒否されます。

**def_data.lu_model**

LU のモデル・タイプと番号。このフィールドは、従属型 LU によってのみH用され、独立型 LU の場合は AP_UNKNOWN に設定する , 要があります。従属型 LU の場合には、以下のいずれかの値に設定されます。

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3

AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

従属型 LU に対して **model_name** がすべて 2 進ゼロにセットされている場合には、このフィールドは無 k されます。AP_UNKNOWN 以 0 の値が X 定され、ホスト・システムが SDDLU（自己定義従属 LU）をサポートしている場合、ノードは非送信請求 PSID NMVT 応答を作成して、ホストにあるローカル LU を動的に定義します。 PSID サブベクトルはこのフィールドの値に対応したマシン番号 (マシン・タイプ)と型式番号を含みます。このフィールドは verb の再発行により動的に変更されることもあります。変更は、LU がクローズされて非 h 動化されたときまでは有効になりません。

**def_data.model_name**

LU のモデル名。このフィールドは、従属型 LU によってのみ H 用され、独立型 LU の場合は 2 進ゼロに設定する , 要があります。APPN はこのフィールドが EBCDIC 文 z の A-Z、0-9、@、# および $ から構成されているか検査します。

このフィールドが 2 進ゼロにセットされていなくて、かつホスト・システムが SDDLU をサポートしている場合には、ホストでローカル LU を動的に定義するためにノードは非送信請求 PSID NMVT 応答を生成します。 PSID サブベクトルははこのフィールドに提供された名前が含まれています。 **def_data.model_name** は verb の再発行により動的に変更できます。変更は、LU がクローズされて非 h 動化されたときまでは有効になりません。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_LU_MODEL

AP_INVALID_LU_NAME
AP_INVALID_NAU_ADDRESS
AP_INVALID_SESSION_LIMIT
AP_INVALID_DISABLE

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PU_NOT_DEFINED

AP_INVALID_LU_NAME
AP_LU_ALREADY_DEFINED
AP_ALLOCATE_NOT_PENDING
AP_LU_ALIAS_ALREADY_USED
AP_PLU_ALIAS_ALREADY_USED
AP_PLU_ALIAS_CANT_BE_CHANGED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

**secondary_rc**
AP_MEMORY_SHORTAGE

# DEFINE_LS

DEFINE_LS をH用して、新しいリンク・ステーション (LS) を定義するか、既存のリンク・ステーションを修正します。この verb により、ノード内で固有な LS 名と、この LS がH用するポートの名前がUけられます。このポートは、すでに DEFINE_PORT verb をH用して定義されていなければなりません。リンク特有のデータは、基本構造体に連結されます。 (STOP_LS の発行後に）リンク・ステーションがリセット状態である場合、DEFINE_LS だけをH用して既存のリンク・ステーションのフィールドを 1 つ以上修正することができます。この DEFINE_LS でX定された **port_name** は、以前に LS を定義して以来変更されていません。

DLC、ポート、およびリンク・ステーションの関連についての詳細は、15ページの『DLC プロセス、ポート、リンク・ステーション』を2照してください。

**LS_DEF_DATA** での様々なフィールドの設定は、**adj_cp_type** フィールドの値によって異なってきます。 **adj_cp_type** がH用することのできる値 (**def_data.adj_cp_type** の t 分で詳細を説明) は以下のように 8 個ありますが、そのうちの 4 つは、隣接タイプ 2.1 (APPN) ノードへのリンクのためにH用されます。

- AP_NETWORK_NODE
- AP_END_NODE
- AP_APPN_NODE
- AP_BACK_LEVEL_LEN_NODE

以下の 4 つは、PU タイプ 2.0 通信だけを伝送するリンクのためにH用されます。

- AP_HOST_XID3
- AP_HOST_XID0
- AP_DSPU_XID
- AP_DSPU_NOXID

APPN ノードには 4 種類あり、それぞれは以下のようにして区別します。

- APPN ネットワーク・ノードは、XID3 内にネットワーク名制御ベクトル (CV) を含み、並列 TG をサポートし、XID3 内にネットワーキング機能ビットを設定し、そしてリンク上で CP-CP セッションをサポートします。
- APPN エンド・ノードは、XID3 内にネットワーク名 CV を含み、並列 TG をサポートし、XID3 内にネットワーキング機能ビットを設定せず、そしてリンク上で CP-CP セッションをサポートします。
- アップ・レベル・ノードは、XID3 内にネットワーク名 CV を含み、並列 TG をサポートし、XID3 内にネットワーキング機能ビットを設定せず、そして CP-CP セッションをサポートしません。
- バック・レベル・ノードは、XID3 内にネットワーク名 CV を含まず、並列 TG をサポートせず、XID3 内にネットワーキング機能ビットを設定せず、そして CP-CP セッションをサポートしません。

以下のフィールドは、すべてのリンク向けに設定されなければなりません。

port_name

adj_cp_type

dest_address

auto_act_supp

disable_remote_act

limited_resource

link_deact_timer

ls_attributes

adj_node_id

local_node_id

target_pacing_count

max_send_btu_size

link_spec_data_len

ls_role

その他のフィールドは、以下のように設定しなければなりません。

- **adj_cp_type** が AP_NETWORK_NODE、AP_END_NODE、または AP_APPN_NODE に設定されている場合、以下のフィールドを設定しなければなりません。

  adj_cp_name

  tg_number

  solicit_sscp_sessions

  dspu_services

  hpr_supported

  hpr_link_lvl_error

  default_nn_server

  cp_cp_sess_support

  use_default_tg_chars

  tg_chars

- **adj_cp_type** が AP_BACK_LEVEL_LEN_NODE に設定されている場合、以下のフィールドを設定しなければなりません。

  adj_cp_name

  solicit_sscp_sessions

  dspu_services

  use_default_tg_chars

  tg_chars

- ローカル PU がリンクを H 用する場合（**adj_cp_type** が AP_HOST_XID3 か AP_HOST_XID0 に設定されている、または **solicit_sscp_sessions** が APPN ノードへのリンク上で AP_YES に設定されている）、以下のフィールドを設定しなければなりません。

  **pu_name**

- ダウンストリーム PU がリンクを H 用し、PU 集信によって H 用される場合（**dspu_services** が AP_PU_CONCENTRATION に設定されている）、以下のフィールドを設定しなければなりません。

  **dspu_name**

- ダウンストリーム PU がリンクをH用し、DLUR によってH用される場合（**dspu_services** が AP_DLUR に設定されている）、以下のフィールドを設定しなければなりません。

  dspu_name

  dlus_name

  bkup_dlus_name

## VCB 構造体

```
typedef struct define_ls
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* current format is zero        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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[RD_LEN]; /* resource description          */
        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 flag */
        unsigned char   dspu_services;       /* Services provided for         */
                                             /* downstream PU                 */
        unsigned char   dspu_name[8];        /* Downstream PU name (reserved  */
                                             /* if dspu_services is set to    */
                                             /* AP_NONE or AP_DLUR)           */
        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 link use link-level      */
                                             /* error recovery for HPR frms?  */
        unsigned short  link_deact_timer;    /* HPR link deactivation timer   */
        unsigned char   reserv1;             /* reserved                      */
        unsigned char   default_nn_server;   /* Use as deflt 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            */
        unsigned short  target_pacing_count; /* target pacing count           */
        unsigned short  max_send_btu_size;   /* max send BTU size */
        unsigned char   ls_role;             /* link station role to use      */
                                             /* on this link                  */
        unsigned char   max_ifrm_rcvd;       /* max number of I-frames rcvd   */
```

```
        unsigned short  dlus_retry_timeout;  /* DLUS retry timeout            */
        unsigned short  dlus_retry_limit;    /* DLUS retry limit              */
        unsigned char   conventional_lu_compression;
                                             /* Data compression requested for */
                                             /* conventional LU sessions      */
        unsigned char   conventional_lu_cryptography;
                                             /* Cryptography required for     */
                                             /* conventional LU sessions      */
        unsigned char   reserv3;             /* reserved                      */
        unsigned char   retry_flags;         /* conditions LU sessions        */
        unsigned short  max_activation_attempts;
                                             /* how many automatic retries:   */
        unsigned short  activation_delay_timer;
                                             /* delay between automatic retries*/
        unsigned char   branch_link_type;    /* branch link type              */
        unsigned char   adj_brn_cp_support;  /* adjacent BrNN CP support      */
        unsigned char   reserv4[20];         /* reserved                      */
        unsigned short  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;         /* modem class                   */
        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 short  length;              /* length                        */
        unsigned short  reserve1;            /* reserved                      */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                             /* address                       */
} LINK_ADDRESS;

typedef struct link_spec_data
{
       unsigned char  link_data[SIZEOF_LINK_SPEC_DATA];

}  LINK_SPEC_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_LS

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**ls_name**

リンク・ステーションの名前。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。

フィールド **ls_name** を特殊値「$ANYNET$」（ASCII ストリング）に設定すると、これは AnyNet DLC によって経路X定される独立 LU セッション通信量の送信先のリンク・ステーションである、ということがノード・オペレーター機能に通知されるようになります。この名前のリンク・ステーションは、AnyNet 経路X定が , 要であれば、AnyNet DLC を介したポート上で定義されなければなりません。

**def_data.description**

q 源の説明（QUERY_LS、QUERY_PU で戻される）。これは、ローカルに=示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**def_data.port_name**

このリンク・ステーションと関連したポートの名前。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。名前のついたポートは、DEFINE_PORT verb によって定義されている , 要があります。

**def_data.adj_cp_name**

完全修飾 17 バイト隣接制御点名。右側には EBCDIC スペースを埋め込みます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。 (それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。) このフィールドは、APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。 APPN ノードへのリンクである場合、フィールド **tg_number** が 1 から 20 までの範囲の数に設定されない限り、もしくはフィールド **adj_cp_type** が AP_BACK_LEVEL_LEN_NODE に設定されない限りすべてゼロに設定できます。すべてゼロに設定されると、XID 交換~に隣接ノードから受け取る名前については検査されません。すべてゼロに設定されない場合で、**adj_cp_type** が AP_BACK_LEVEL_LEN_NODE に設定されない限り、 XID 交換~に隣接ノードから受け取る名前について検査されます（この場合隣接ノードを識別するのにH用）。

**def_data.adj_cp_type**

隣接ノードのタイプ。

**AP_NETWORK_NODE**

ノードが APPN ネットワーク・ノードであることを示します。

**AP_END_NODE**

このノードが APPN エンド・ノードまたはアップ・レベル・ノードであることをX定します。

**AP_APPN_NODE**

このノードが APPN ネットワーク・ノード、APPN エンド・ノード、

またはアップ・レベル・ノードであることをX定します。ノード・タイプは、XID の交換~にN認されます。

**AP_BACK_LEVEL_LEN_NODE**
このノードが back_level_len ノードであることをX定します。つまり、XID で制御点名を送信しないということです。独立 LU セッションをサポートする AnyNet DLC をH用したリンクの場合、AP_BACK_LEVEL_LEN_NODE をX定する，要があります。

**AP_HOST_XID3**
このノードがホストであり、パーソナル・コミュニケーションズまたは Communications Serverは形式が 3 XID であるノードからのポーリング XID に応答することをX定します。

**AP_HOST_XID0**
このノードがホストであり、パーソナル・コミュニケーションズまたは Communications Serverは形式が 0 XID であるノードからのポーリング XID に応答することをX定します。従属 LU セッションをサポートする AnyNet DLC をH用したリンクの場合、AP_HOST_XID0 をX定する，要があります。

**AP_DSPU_XID**
このノードがダウンストリーム PU であり、パーソナル・コミュニケーションズまたは Communications Server はリンクをh動化するときに XID の交換を組み込むことをX定します。

**AP_DSPU_NOXID**
このノードがダウンストリーム PU であり、パーソナル・コミュニケーションズまたは Communications Server はリンクをh動化するときに XID の交換を組み込まないことをX定します。

**注:** VRN へのリンク・ステーションは常に動的であるため、定義されません。

**def_data.dest_address.length**
隣接ノードでの宛先リンク・ステーションのアドレスの長さ。

**def_data.dest_address.length** がゼロにセットされ、かつこの LS が タイプ SATF のポートと関連している場合には、「プログラム」はこのリンク・ステーションをワイルド・カード・リンク・ステーションとみなします。これは「プログラム」が LS をどの着信接続に対してもマッチングさせる原因となります。LS は定義された他のリンク・ステーションによってはマッチングされません。

**def_data.dest_address.address**
隣接ノードでのリンク・ステーションの宛先アドレス。 AnyNet DLC をH用したリンクの場合、**dest_address** は隣接ノード ID または隣接制御点名をX定します。隣接ノード ID をX定する場合、長さは 4 とし、アドレスには 4 バイトの 16 進ノード ID（1 バイトのブロック ID、3 バイトの PU ID）が含まれていなければなりません。隣接制御点名をX定する場合は、長さは 17 であり、アドレスには、 EBCDIC ブランクが埋め込まれた EBCDIC による制御点名が含まれていなければなりません。

**def_data.auto_act_supp**

セッションが , 要とする場合に、リンクを自動的に h 動化できるかどうかを X 定します（AP_YES または AP_NO)。このリンクが APPN ノードへのリンクでない場合、このフィールドは常に AP_YES に設定され、その他のパラメーターについての要件はありません。このリンクが APPN ノードへのリンクである場合、リンクが CP-CP セッションもサポートしていれば、このフィールドを AP_YES に設定することはできません。 v 前に d り当てられた TG の番号がリンク **tg_number** のために定義されており、かつ 1 から 20 の範囲の値に設定されている場合に限り、AP_YES に設定できます。この場合、**cp_cp_sess_support** および **tg_number** は無 k されるため、 **adj_cp_type** が AP_BACK_LEVEL_LEN_NODE に設定されていれば、この要件は , ず満たされます。

**def_data.tg_number**

v 前に d り当てた TG の番号。このフィールドは、このリンクが隣接 APPN ノードへのリンクである場合にのみ適切であり、それ以 0 の場合には無 k されます。**adj_cp_type** が AP_BACK_LEVEL_LEN_NODE に設定される場合も、このフィールドは無 k され、1 に設定されたものと見なされます。隣接 APPN ノードへのリンクでは、これは 1 から 20 までの範囲の値に設定されなければなりません。この番号を H 用して、リンクが h 動化されたときのリンクを示します。このリンクを h 動化するときには、パーソナル・コミュニケーションズまたは Communications Server は隣接ノードからのその他の番号は受け入れません。TG の番号を v 前に d り当てるときに T 一致が生じてリンクの h 動化が失敗してしまうのを防ぐため、同じ TG の番号を隣接リンク・ステーションの隣接ノードによって定義する , 要があります ( v 前に d り当てた TG の番号を H 用する場合)。 v 前に d り当てた TG の番号を定義しているのであれば、 **adj_cp_name** も定義し (ただしすべてゼロに設定することはできない)、**adj_cp_type** を AP_NETWORK_NODE または AP_END_NODE に設定しなければなりません。ゼロを入力した場合、TG 番号は v 前に d り当てられず、リンクの h 動化 ~ の交渉によって決定されます。

**def_data.limited_resource**

リンクを H 用しているセッションがない場合、リンク・ステーションが非 h 動化されるかどうかを示します。これは、以下の値の 1 つに設定されます。

**AP_NO**

このリンクは限定された q 源ではなく、自動的に非 h 動化されることはありません。

**AP_YES または AP_NO_SESSIONS**

このリンクは限定された q 源であり、このリンクを H 用している h 動セッションがない場合に自動的に非 h 動化されます。限定された q 源のリンク・ステーションは、CP-CP セッション・サポート用に構成することができます。 (これは、このフィールドを AP_YES に設定し、**cp_cp_sess_support** を AP_YES に設定することにより実行できます。) この場合、CP-CP セッションがリンクを介して大きくなっていくと、パーソナル・コミュニケーションズまたは Communications Server はリンクを限定された q 源として処理しなくなります (そしてリンクを小さくすることもありません)。

**AP_INACTIVITY**
このリンクは限定されたq源であり、このリンクをH用しているh動セッションがない場合、もしくは **link_deact_timer** フィールドでX定した間にリンク上でデータのフローがない場合、自動的に非h動化されます。非交換ポート上のリンク・ステーションは限定されたq源として構成できないことに注意してください。

非交換ポート上のリンク・ステーションは限定されたq源として構成できないことに注意してください。

限定されたq源のリンク・ステーションは、CP-CP セッション・サポート用に構成することができます。(これは、このフィールドを AP_YES に設定し、**cp_cp_sess_support** を AP_YES に設定することにより実行できます。) この場合、 CP-CP セッションがリンクを介して大きくなっていくと、パーソナル・コミュニケーションズまたは Communications Server はリンクを限定されたq源として処理しなくなります (そしてリンクを小さくすることもありません)。これが AP_INACTIVITY にセットされている場合には、これは適用しないことに注意してください。

**def_data.solicit_sscp_sessions**
AP_YES をX定すると、SSCP と従属 LU の間のセッションおよびローカル制御点と従属 LU の間のセッションを+ Oするよう隣接ノードに要求が送られます。(この場合、**pu_name** を設定する , 要があります。) AP_NO をX定すると、このリンクでは SSCP とのセッションは要求されません。このフィールドは、このリンクが APPN ノードへのリンクである場合にのみ適切であり、それ以 0 の場合には無k されます。隣接ノードがホストとなるように定義されている場合 (**adj_cp_type** が AP_HOST_XID3 または AP_HOST_XID0 に設定されている) 、パーソナル・コミュニケーションズまたは Communications Server は SSCP と従属 LU の間のセッションおよびローカル制御点と従属 LU の間のセッションを+ Oするようにそのホストに常に要求します (もう一度 **pu_name** を設定する , 要がある)。

このフィールドは、**dspu_services** が AP_NONE にセットされている場合には、隣接 APPN ノードへのリンクに関しては AP_YES にだけセットできます。このフィールドが AP_YES にセットされ、かつ LS がH用する DCL が hpr_only として定義されている場合には、DEFINE_LS はパラメーター検査と AP_INVALID_SOLICIT_SSCP_SESS の 2 次戻りコードでリジェクトされます。

**def_data.pu_name**
隣接ノードがホストとなるように定義されているか、APPN ノードへのリンクで **solicit_sscp_sessions** が AP_YES に設定されている場合に、このリンクをH用するローカル PU の名前。これは、8 バイト、英数z、タイプ A の EBCDIC 文z列 (先頭は文z) です。EBCDIC スペースが右の余白に埋め込まれます。隣接ノードがホストとなるように定義されておらず、**solicit_sscp_sessions** が AP_YES に設定されている APPN ノードとして定義されていない場合、このフィールドは無k されます。

**def_data.disable_remote_act**
このリンクのリモートh動化がサポートされるかどうかを示します (AP_YES または AP_NO)。

**def_data.dspu_services**

このリンクを介してローカル・ノードがダウンストリーム PU に提供するサービスをX定します。これは、以下の 1 つに設定されます。

**AP_PU_CONCENTRATION**

ローカル・ノードは、ダウンストリーム PU に PU 集信を提供します。

**AP_DLUR**

ローカル・ノードは、ダウンストリーム PU に DLUR サービスを提供します。この設定は、ローカル・ノードがネットワーク・ノードの場合のみ有効です。

**AP_NONE**

ローカル・ノードは、このダウンストリーム PU にサービスを提供しません。

このフィールドが AP_PU_CONCENTRATION または AP_DLUR に設定される場合、**dspu_name** も設定する , 要があります。

隣接ノードがダウンストリーム PU として定義されている場合（つまり、**adj_cp_type** が AP_DSPU_XID または AP_DSPU_NOXID に設定されている）、このフィールドは AP_PU_CONCENTRATION または AP_DLUR に設定しなければなりません。 **solicit_sscp_sessions** が AP_NO に設定されている場合、APPN ノードへのリンクでは AP_PU_CONCENTRATION または AP_DLUR に設定できます。隣接ノードがホストとして定義されている場合は、このフィールドは無 k されます。

このフィールドが AP_NONE にセットされていなくて、かつ LS が H用する DLC が hpr_only として定義されている場合には、DEFINE_LS はパラメーター検査と SP_INVALID_DSPU_SERVICES の 2 次戻りコードでリジェクトされます。

**def_data.dspu_name**

ダウンストリーム PU の名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**dspu_services** が AP_PU_CONCENTRATION または AP_DLUR に設定されている場合には、このフィールドを設定する , 要があり、設定しない場合には無 k されます。

**def_data.dlus_name**

ダウンストリーム・ノードへのリンクがh動化されるときに、DLUR が SSCP サービスを送信請求する DLUS ノードの名前。この名前は、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文 z ストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。）このフィールドがすべてゼロに設定されている場合、このグローバルなデフォルト DLUS（DEFINE_DLUR_DEFAULTS verb をH用して定義されている場合）は、リンクがh動化されるときに送信請求されます。 **dlus_name** がゼロに設定され、かつグローバルなデフォルト DLUS がない場合、リンクがh動化

されるときに DLUR は SSCP の接続を + O しません。 **dspu_services** が AP_DLUR に設定されていなければ、このフィールドは無 k されます。

**def_data.bkup_dlus_name**
ダウンストリーム PU のバックアップとして H 用される DLUS ノードの名前。この名前は、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文 z ストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。) このフィールドがすべてゼロに設定されている場合、グローバルなバックアップ・デフォルト DLUS (DEFINE_DLUR_DEFAULTS verb を H 用して定義されている場合) は、この PU のバックアップとして H 用されます。 **dspu_services** が AP_DLUR に設定されていなければ、このフィールドは無 k されます。

**def_data.hpr_supported**
このリンク上で HPR をサポートするかどうかを X 定します (AP_YES または AP_NO)。このフィールドは、このリンクが APPN ノードへのリンクである場合にのみ適切であり、それ以 0 の場合には無 k されます。リンクが APPN ノードへのリンクではない場合には、このフィールドを AP_YES に設定すると verb がパラメーター検査と INVALID_NODE_TYPE_FOR_HPR の 2 次戻りコードでリジェクトされる結果となります。

**def_data.hpr_link_lvl_error**
リンク・レベルのエラー回 | を H 用してこのリンク上で HPR 通信を送信するかどうかを X 定します (AP_YES または AP_NO)。 **hpr_supported** が AP_NO に設定されていると、このパラメーターは無 k されます。

**def_data.link_deact_timer**
限度のある q 源のリンク非 h 動化タイマー (C 数)。

**limited_resource** が AP_INACTIVITY に設定される場合で、このタイマーの X 定期間内にデータがリンクを走査しない場合は、リンクは自動的に非 h 動化されます。

ゼロを X 定する場合、デフォルト値として 30 C が H 用されます。それ以 0 の場合、最小値は 5 です。(これより小さい値を X 定しても、X 定した値は無 k され、5 が H 用されます。) **limited_resource** が AP_NO に設定されていると、このパラメーターは予約されます。

**def_data.default_nn_server**
ネットワーク・ノード・サーバーへの CP-CP セッションをサポートするために、エンド・ノードによってリンクが自動的に h 動化されるかどうかを X 定します (AP_YES または AP_NO)。このフィールドを有効にするには、このリンクを定義して、CP-CP セッションをサポートしなければならないことに注意してください。

**def_data.ls_attributes**
隣接ノードについてさらに情報を X 定します。

**def_data.ls_attributes[0]**
ホスト・タイプ。

**AP_SNA**
8 準 SNA ホスト。

**AP_FNA**
FNA (VTAM-F) ホスト。

**AP_HNA**
HNA ホスト。

**def_data.ls_attributes[1]**
このフィールドはビット・フィールドです。それは値 AP_NO をとるか、またはビット単位で互いに OR 結合された以下の値のいずれでも取ります。

**AP_SUPPRESS_CP_NAME**
バック・レベル LEN ノードへのリンクのためのネットワーク名 CV 抑 _ オプション。このビットが設定されている場合には、ネットワーク名 CV は隣接ノードとの XID 交換に組み込まれます。(**adj_cp_type** を AP_BACK_LEVEL_LEN_NODE または AP_HOST_XID3 に設定しないと、このビットは無 k されます)。

**AP_REACTIVATE_ON_FAILURE**
リンクがアクティブであり、そのときに障 2 が起これば、パーソナル・コミュニケーションズまたは Communications Serverはリンクの再 h 動化を n 行します。再 h 動化が失敗した場合には、リンクは非アクティブのままとなります。

**AP_USE_PU_NAME_IN_XID_CVS**
隣接ノードがホストであると定義されるか、または **solicit_sscp_sessions** が APPN ノードへのリンクで tp AP_YES とセットされている、かつ AP_SUPPRESS_CP_NAME ビットがセットされていない場合には、 Format 3 XID で送信されたネットワーク名 CV の完全修飾 CP 名は **def_data.pu_name** で提供され、CP のネットワーク ID で完全修飾された名前で置換されます。

**def_data.adj_node_id**
隣接ノードのノード ID。これは、4 バイトの16 進数文 z 列です。**adj_cp_type** で、隣接ノードが T2.1 ノードであることが示される場合、これがゼロ以 0 でない限りこのフィールドは無 k され、**adj_cp_type** が AP_BACK_LEVEL_LEN_NODE に設定されるか、隣接ノードはネットワーク名 CV を XID3 に送信しないかのいずれかです。**adj_cp_type** が AP_HOST_XID3 または AP_HOST_XID0 に設定される場合、このフィールドは常に無 k されます。 **adj_cp_type** が AP_DSPU_XID に設定され、このフィールドがゼロ以 0 である場合、このフィールドはダウンストリーム PU の ID を検査するのに H 用されます。 **adj_cp_type** が AP_DSPU_NOXID に設定される場合、このフィールドは無 k される（**dspu_services** が AP_PU_CONCENTRATION の場合）か、 DLUS へのダウンストリーム PU を識別するのに H 用される（**dspu_services** が AP_DLUR の場合）かのいずれかです。

**def_data.local_node_id**
このリンク・ステーション上の XID に送信されたノード ID。これは、4 バイトの16 進数文 z 列です。このフィールドがゼロに設定される場合、XID 交換 ~ には **node_id** が H 用されます。このフィールドがゼロ以 0 であれば、この LS 上で XID 交換 ~ の値が置き換えられます。

**def_data.cp_cp_sess_support**

CP-CP セッションをサポートするかどうかをX定します（AP_YES または AP_NO）。このフィールドは、このリンクが APPN ノードへのリンクである場合にのみ適切であり、それ以 0 の場合には無 k されます。 **adj_cp_type** が AP_BACK_LEVEL_LEN_NODE に設定される場合も、このフィールドは無 k され、AP_NO に設定されたものと見なされます。

**def_data.use_default_tg_chars**

DEFINE_PORT verb によって提供されているデフォルトの TG 特性をH用するかどうかをX定します（AP_YES または AP_NO）。AP_YES に設定される場合、**tg_chars** フィールドは無 k されます。このフィールドは、このリンクが APPN ノードへのリンクである場合にのみ適切であり、それ以 0 の場合には無 k されます。

**def_data.tg_chars**

TG 特性（33ページの『DEFINE_CN』を 2 照）。このフィールドは、このリンクが APPN ノードへのリンクである場合にのみ適切であり、それ以 0 の場合には無 k されます。

**def_data.target_pacing_count**

1 から 32 767 の数値であり、この TG での BIND 用の望ましいペーシング・ウィンドウ・サイズを包 g 的に示します。この番号は、固定バインド・ペーシングの実行 ~ にのみ有効です。パーソナル・コミュニケーションズまたは Communications Server は現在この値をH用していません。

**def_data.max_send_btu_size**

このリンク・ステーションから送信できる BTU サイズの最大値。この値をH用して、1 組のリンク・ステーション間で伝送できる BTU サイズの最大値を決定します。リンクに HPR 機能がない場合、この値は 99 以上に設定しなければなりません。リンクに HPR 機能がある場合、この値は 768 以上に設定する , 要があります。

**def_data.ls_role**

このリンク・ステーションが想定するリンク・ステーション・ロール。これは、折衝可能な 1 次または 2 次のロールを選択するために、AP_LS_NEG、AP_LS_PRI、または AP_LS_SEC のいずれかになります。また、このフィールドに AP_USE_PORT_DEFAULTS を設定して、DEFINE_PORT verb に構成された値を選択できます。**dlc_type** が AP_TWINAX であれば、 AP_LS_SEC だけがサポートされます。**dlc_type** が AP_ANYNET（および **ls_name** が “$ANYNET$”）であれば、AP_LS_PRI はサポートされません。

**def_data.max_ifrm_rcvd**

肯定応答の前に XID 送信側が受け取れる I フレームの最大数。

範囲: 0 - 127

ゼロがX定された場合には、DEFINE_PORT からの **max_ifrm_rcvd** の値がデフォルトとしてH用されます。

**def_data.dlus_retry_timeout**

**def_data.dlus_name** と **def_data.bkup_dlus_name** フィールドにX定された DLUS への接続を 2 回以上 n みるときの間V（C数）。最初の n みと最初の再 n 行の間の間Vは、常に 1 Cです。ゼロをX定すると、

DEFINE_DLUR_DEFAULTS を介して構成されたデフォルト値がH用されます。**def_data.dspu_services** が AP_DLUR に設定されていなければ、このフィールドは無 k されます。

**def_data.dlus_retry_limit**

**def_data.dlus_name** と **def_data.bkup_dlus_name** フィールドにX定された DLUS への接続が最初に失敗した後で再 n 行する最大の回数。ゼロをX定すると、DEFINE_DLUR_DEFAULTS を介して構成されたデフォルト値がH用されます。X'FFFF' がX定されると、APPN は無限に再 n 行します。**def_data.dspu_services** が AP_DLUR に設定されていなければ、このフィールドは無 k されます。

**def_data.conventional_lu_compression**

データ圧縮をこの PU に依存する在来の LU セッションに要求するかどうかをX定します。このフィールドは LU 0 から 3 のトラフィックを運ぶリンクにのみ有効であることに注意してください。

**AP_NO**

ローカル・ノードは、この PU をH用する在来の LU セッションでのデータ・フローを圧縮も圧縮解除もすべきではありません。

**AP_YES**

データ圧縮は、ホストが圧縮を要求している場合には、この PU に依存する LU セッションにH用可能化されるべきです。この値がセットされているが、ノードが圧縮 ( START_NODE verb で定義された) をサポートしない場合には、リンク・ステーションは正常に定義されますが、圧縮サポートはありません。

象となる条件をX定します。それはビット・フィールドで、以下の値をビット単位で互いに OR 結合したいずれのものでも可能です。

**AP_RETRY_ON_START**
h 動化が n 行されたときにリモート・ノードから無応答の場合には、リンク h 動化が再 n 行されます。 h 動化が n 行されたときに基になっているポートが非 h 動である場合には、「プログラム」はそれを h 動化しようとします。

**AP_RETRY_ON_FAILURE**
リンクが h 動中、または h 動保留 ~ に障 2 があれば、リンク h 動化が再 n 行されます。 h 動化が n 行されたときに基になっているポートに障 2 があれば、「プログラム」はそれを h 動化しようとします。

**AP_RETRY_ON_DISCONNECT**
リンクがリモート・ノードにより正規に停 _ されれば、リンク h 動化が再 n 行されます。

**AP_DELAY_APPLICATION_RETRIES**
リンク h 動化は、アプリケーション (START_LS またはオンデマンド・リンク h 動化を H 用したもの) により + O されて、**activation_delay_timer** をH用してペースを取られます。

**AP_INHERIT_RETRY**
このフィールドのフラグがX定する再 n 行条件に加えて、基になっているポート定義の **retry_flags** フィールドでX定したこれらのものもH用されます。

**def_data.max_activation_attempts**
このフィールドは、少なくとも 1 つのフラグが **retry_flags**にセットされていなければ効力がありません。

このフィールドは、リモート・ノードが応答しないか、または基になっているポートが非 h 動であるときに、「プログラム」 が許可する再 n 行の数をX定します。これには自動的再 n 行とアプリケーション主導型の h 動化 n 行が含まれます。

この限&に達すると、それ以上の自動的再 n 行は行われません。この条件は STOP_LS、STOP_PORT、STOP_DLC または h 動化の成功によりリセットされます。 START_LS または OPEN_LU_SSCP_SEC_RQ は 1 回 だけの h 動化 n 行に終わります、 h 動化が失敗しても再 n 行は行われません。

ゼロは ’限&がない’ ことを意味します。値 AP_USE_DEFAULTS は DEFINE_PORT でシステムに提供された **max_activiation_attempts** のH用という結果になります。

**def_data.activation_delay_timer**
このフィールドは、少なくとも 1 つのフラグが **retry_flags**にセットされていなければ効力がありません。

このフィールドは、AP_DELAY_APPLICATION_RETRIES ビットが **def_data.retry_flags** にセットされている場合に、自動的再 n 行間に、およびアプリケーション主導型 h 動化 n 行間に 「プログラム」 が待つC数をX定します。

値 AP_USE_DEFAULTS は DEFINE_PORT でシステムに提供された **activiation_delay_timer** のH用という結果になります。

ゼロがX定された場合には、「プログラム」は 30 Cのデフォルト・タイマー~間をH用します。

**def_data.branch_link_type**

BrNN のみ。これはリンクがアップリンクかまたはダウンリンクかをX定します。このフィールドは、**def_data.adj_cp_type** が AP_NETWORK_NODE、AP_END_NODE、AP_APPN_NODE、または AP_BACK_LEVEL_LEN_NODE にセットされている場合にのみ適用します。

**AP_UPLINK**

このリンクはアップリンクです。

**AP_DOWNLINK**

このリンクはダウンリンクです。

フィールド **adj_cp_type** が AP_NETWORK_NODE にセットされている場合には、このフィールドは AP_UPLINK にセットされる , 要があります。

他のノード・タイプ : このフィールドは無 k されます。

**def_data.adj_brnn_cp_support**

BrNN のみ。これは、隣接 CP が許されているか、要件であるか、あるいは NN (BrNN) であることが禁 _ されているかどうかをX定します。たとえば、BrNN は NN 0 観を示しています。このフィールドは、フィールド **adj_cp_type** が AP_NETWORK_NODE または AP_APPN_NODE にセットされている場合 (そして XID 交換の間にN認されたノード・タイプがネットワーク・ノードである場合) にのみ適用されます。

**AP_BRNN_ALLOWED**

隣接 CP は NN (BrNN) であることを許可されます (しかし、 , 要ではありません)。

**AP_BRNN_REQUIRED**

隣接 CP は NN (BrNN) である , 要があります。

**AP_BRNN_PROHIBITED**

隣接 CP は NN (BrNN) であることを許可されません。

フィールド **adj_cp_type** が AP_NETWORK_NODE にセットされ、かつフィールド **auto_act_supp** が AP_YES にセットされている場合には、このフィールドは AP_BRNN_REQUIRED または AP_BRNN_PROHIBITED にセットされる , 要があります。

他のノード・タイプ : このフィールドは無 k されます。

**def_data.link_spec_data_len**

このフィールドは、常にゼロに設定されていなければなりません。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_DEF_LINK_INVALID_SECURITY

AP_INVALID_CP_NAME
AP_INVALID_LIMITED_RESOURCE
AP_INVALID_LINK_NAME
AP_INVALID_LS_ROLE
AP_INVALID_NODE_TYPE
AP_INVALID_PORT_NAME
AP_INVALID_AUTO_ACT_SUPP
AP_INVALID_PU_NAME
AP_INVALID_SOLICIT_SSCP_SESS
AP_INVALID_DLUS_NAME
AP_INVALID_BKUP_DLUS_NAME
AP_INVALID_NODE_TYPE_FOR_HPR
AP_INVALID_TARGET_PACING_COUNT
AP_INVALID_BTU_SIZE
AP_HPR_NOT_SUPPORTED
AP_INVALID_TG_NUMBER
AP_MISSING_CP_NAME
AP_MISSING_CP_TYPE
AP_MISSING_TG_NUMBER
AP_PARALLEL_TGS_NOT_SUPPORTED
AP_INVALID_DLUS_RETRY_TIMEOUT
AP_INVALID_DLUS_RETRY_LIMIT
AP_INVALID_CLU_CRYPTOGRAPHY
AP_INVALID_RETRY_FLAGS
AP_BRNN_SUPPORT_MISSING
AP_INVALID_BRANCH_LINK_TYPE
AP_INVALID_BRNN_SUPPORT

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_LOCAL_CP_NAME

AP_DEPENDENT_LU_SUPPORTED
AP_DUPLICATE_DEST_ADDR

AP_INVALID_NUM_LS_SPECIFIED
AP_LS_ACTIVE
AP_PU_ALREADY_DEFINED
AP_DSPU_SERVICES_NOT_SUPPORTED
AP_DUPLICATE_TG_NUMBER
AP_TG_NUMBER_IN_USE
AP_CANT_MODIFY_VISIBILITY
AP_INVALID_UPLINK
AP_INVALID_DPWNLINK

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_LU_0_TO_3

この verb は、タイプ 0、1、2 または 3 の LU を定義します。これにより、LU を LU プールに追加できるようになります。プールがまだ存在していない場合には、追加されます。この verb は既存定義の **lu_model**、**model_name**、**priority**、**description**、および**appc_spec_def_data**を変更するためにはH用できません。しかし他のフィールドは変更できます。

パーソナル・コミュニケーションズまたは Communications Server は、ACTLU による暗黙の LU タイプ 0、 1、2 または 3 定義をサポートします。暗黙定義は削除できませんが、LU が非 h 動化されると除去されます。暗黙定義の詳細が , 要であれば、QUERY_LU_0_TO_3 をH用するか、LU_0_TO_3_INDICATION のレジスターをH用してください。 **lu_name**、**pu_name**、および **nau_address** が正N であり、かつ **pool_name** がすべてゼロであれば、 DEFINE_LU_0_TO_3 をH用して暗黙の LU 定義を再定義できます（その後、この LU は、最初のオペレーターによって構成されていたかのように処理されます)。

## VCB 構造体

### Format 1

```
typedef struct define_lu_0_to_3
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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          /* resource description          */
        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                      */
        unsigned char   sscp_id[6]           /* SSCP ID                       */
        unsigned short  timeout;             /* Timeout                       */
        unsigned char   app_spec_def_data[16];
                                             /* Application Specified Data    */
        unsigned char   model_name[7];       /* LU model name for DDDLU       */
        unsigned char   reserv3[17];         /* reserved                      */
} LU_0_TO_3_DEF_DATA;
```

## VCB 構造体

### Format 0

```
typedef struct define_lu_0_to_3
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* attributes                    */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
```

```
        unsigned long   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          /* resource description          */
        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                      */
        unsigned char   sscp_id[6]           /* SSCP ID                       */
        unsigned short  timeout;             /* Timeout                       */
        unsigned char   app_spec_def_data[16];
                                             /* Application Specified Data    */
} LU_0_TO_3_DEF_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_LU_0_TO_3

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンの 1 つをX定するには、このフィールドをゼロまたは 1 に設定します。

**lu_name**

定義するローカル LU の名前。これは、8 バイト、英数z、タイプ A の EBCDIC 文z列（先頭は文z）です。EBCDIC スペースが右の余白に埋め込まれます。

**def_data.description**

q源の説明（QUERY_LU_0_TO_3 で戻される)。これは、ローカルに=示可能な文zセットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**def_data.nau_address**

LU のネットワーク・アドレス単位アドレス。この値の範囲は 1〜255 でなければなりません。

**def_data.pool_name**

この LU が属する LU プールの名前。 これは、8 バイト、英数z、タイプ A の EBCDIC 文z列（先頭は文z）です。EBCDIC スペースが右の余白に埋め込まれます。LU がプールに属していない場合、このフィールドはすべて 2 進ゼロに設定されます。現在プールが存在していなければ、作成されます。

**def_data.pu_name**

この LU がH用する PU の名前（DEFINE_LS verb 上に示されているよう

に)。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**def_data.priority**
ホストへの送信 ~ の LU の優先順位。これは、以下の値の 1 つに設定されます。

AP_NETWORK
AP_HIGH
AP_MEDIUM
AP_LOW

**def_data.lu_model**
LU のモデル・タイプと番号。これは、以下の値の 1 つに設定されます。

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

Format 1 だけにつき、**model_name** がすべて 2 進ゼロにセットされていない限り、このフィールドは無 k されます。

AP_UNKNOWN 以 0 の値が X 定され、ホスト・システムが DDDLU (Dynamic Definition of Dependent LUs; 従属 LU の動的定義) をサポートしている場合、ホストにあるローカル LU を動的に定義するためにノードは非送信請求 PSID NMVT 応答を生成します。 Format 1 だけにつき、PSID サブベクトルはこのフィールドの値に対応したマシン番号 (マシン・タイプ) と型式番号を含みます。このフィールドは verb の再発行により動的に変更されることもあります。変更は、LU が次にクローズされて非 h 動化されるまでは有効になりません。

**def_data.sscp_id**
このフィールドは、この LU を h 動化するために、許可された SSCP の ID を X 定します。それは 6 バイトの 2 進数フィールドです。フィールドが 2 進ゼロにセットされている場合には、LU は SSCP により h 動化されることもあります。

**def_data.timeout**
C 数で X 定された LU のタイムアウト 。タイムアウトが提供され、LU のユーザーが **allow_timeout** を OPEN_LU_SSCP_SEC_RQ (または、PU 集信のケースではダウンストリーム LU に) X 定した場合には、 LU は、PLU-SLU セッションがこの期間非アクティブのまま D っていて以下の条件中の 1 つが保留中であるということが完了してから、非アクティブ化されます。

- セッションが限定 q 源リンクで渡される。
- セッションが再度 H 用される前に別のアプリケーションが LU を H 用したい。

タイムアウトがゼロにセットされている場合には、LU は非 h 動化されません。

**def_data.app_spec_def_data**
アプリケーションX定の定義データ。このフィールドはパーソナル・コミュニケーションズまたは Communications Server によって解釈されませんが、記憶された後 QUERY_LU_0_TO_3 verb によって戻されます。

**def_data.model_name**
パーソナル・コミュニケーションズまたは Communications Server はこのフィールドが EBCDIC 文 z の A-Z、0-9、@、# および $ から構成されているか検査します。このフィールドがすべて 2 進ゼロにセットされてはいなくて、ホスト・システムが DDDLU (従属 LU の動的定義) をサポートしている場合、ホストにあるローカル LU を動的に定義するためにノードは非送信請求 PSID NMVT 応答を生成します。PSID サブベクトルははこのフィールドに提供された名前が含まれています。このフィールドは verb の再発行により動的に変更されることもあります。変更は、LU がクローズされて非 h 動化されるまでは有効になりません。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_INVALID_PU_NAME
AP_INVALID_PU_TYPE
AP_PU_NOT_DEFINED
AP_LU_ALREADY_DEFINED
AP_LU_NAU_ADDR_ALREADY_DEFD
AP_CANT_MODIFY_VISIBILITY

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE
AP_PU_NOT_DEFINED

AP_LU_NAME_POOL_NAME_CLASH
AP_LU_ALREADY_DEFINED
AP_LU_NAU_ADDR_ALREADY_DEFD

従属 LU サポートをH用してシステムが構築されていないために verb が実行されない場合、「プログラム」は次のパラメーターが戻されます。

**primary_rc**
AP_INVALID_VERB

ノードがまだ+Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停_しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_LU_0_TO_3_RANGE

この verb をH用すると、X定した NAU 範囲内で複数の LU を定義できるようになります。このノード・オペレーターが、ベース名と NAU 範囲をX定します。LU 名は、ベース名と NAU アドレスを結合することにより生成されます。 この verb をH用して、既存の定義を修正することはできません。

たとえば、1 から 4 の NAU 範囲と結合された LUNME のベース名では、LU は LUNME001、LUNME002、LUNME003、および LUNME004 として定義されます。5 未満の非埋め込み文 z で構成されたベース名を定義すると、LU 名は 8 未満の非埋め込み文 z で構成されます。そうすると、パーソナル・コミュニケーションズまたは Communications Server は右に文 z を埋め込んで 8 文 z にします。

## VCB 構造体

### Format 1

```
typedef struct define_lu_0_to_3_range
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   attributes;         /* verb attributes               */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   base_name[5];       /* base name                     */
        unsigned char   reserv3;            /* reserved                      */
        unsigned char   description;        /* resource description          */
        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                      */
        unsigned char   sscp_id[6];         /* SSCP ID                       */
        unsigned short  timeout;            /* Timeout                       */
        unsigned char   app_spec_def_data[16];
                                            /* application specified data    */
        unsigned char   model_name[7];      /* LU model name for DDDLU       */
        unsigned char   name_attributes;    /* Attributes of base name       */
        unsigned char   base_number;        /* Base number for LU names      */
        unsigned char   reserv3[15];        /* reserved                      */
} DEFINE_LU_0_TO_3_RANGE;
```

## VCB 構造体

### Format 0

```
typedef struct define_lu_0_to_3_range
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   attributes;         /* verb attributes               */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   base_name[5];       /* base name                     */
        unsigned char   reserv3;            /* reserved                      */
        unsigned char   description;        /* resource description          */
        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                         */
        unsigned char   sscp_id[6];          /* SSCP ID                          */
        unsigned short  timeout;             /* Timeout                          */
        unsigned char   app_spec_def_data;   /* application specified data       */
} DEFINE_LU_0_TO_3_RANGE;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_LU_0_TO_3_RANGE

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンの 1 つをX定するには、このフィールドをゼロまたは 1 に設定します。

**base_name**

ベースの LU 名。これは、5 バイトのタイプ A の EBCDIC ストリング（文ｚで+O）による英数ｚであり、右側には EBCDIC スペースを埋め込みます。このベース名は、3 文ｚのタイプ A の EBCDIC 数値文ｚで構成されて 10 進数の NAU アドレスを示すものであり、これが NAU 範囲内のF LU に追加されます。

これはフィールド **name_attributes** にビットをセットしないフィールドです。ビットを設定するとこのフィールドの意味を変えてしまいます。

**description**

ｑ源の説明（QUERY_LU_0_TO_3 で戻される）。このフィールドの長さは、4 バイトの倍数でなければならず、ゼロであってはなりません。

**min_nau**

この範囲における NAU アドレスの最小値。この値は、包ｇ的に 1 から 255 とすることができます。

**max_nau**

この範囲における NAU アドレスの最大値。この値は、包ｇ的に 1 から 255 とすることができます。

**pool_name**

この LU が属する LU プールの名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。LU がプールに属していない場合、このフィールドはすべて 2 進ゼロに設定されます。

**pu_name**

この LU がH用する PU の名前（DEFINE_LS verb でX定したとおり）。こ

れは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**priority**

ホストへの送信 ~ の LU の優先順位。これは、以下の値の 1 つに設定されます。

AP_NETWORK
AP_HIGH
AP_MEDIUM
AP_LOW

**lu_model**

LU のモデル・タイプと番号。これは、以下の値の 1 つに設定されます。

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

Format 1 だけにつき、**model_name** がすべて 2 進ゼロにセットされていない限り、このフィールドは無 k されます。

AP_UNKNOWN 以 0 の値が X 定され、ホスト・システムが DDDLU (従属 LU の動的定義)をサポートしている場合、ホストにあるローカル LU を動的に定義するためにノードは非送信請求 PSID NMVT 応答を生成します。 Format 1 だけにつき、PSID サブベクトルはこのフィールドの値に対応したマシン番号 (マシン・タイプ) と型式番号を含みます。このフィールドは verb の再発行により動的に変更されることもあります。変更は、LU が次にクローズされて非 h 動化されるまでは有効になりません。

**lu_0_to_3_detail.def_data.sscp_id**

このフィールドは、この LU を h 動化するために、許可された SSCP の ID を X 定します。それは 6 バイトの 2 進数フィールドです。フィールドが 2 進ゼロにセットされている場合には、LU は SSCP により h 動化されることもあります。

**lu_0_to_3_detail.def_data.timeout**

C 数で X 定された LU のタイムアウト。タイムアウトが提供され、LU のユーザーが **allow_timeout** を OPEN_LU_SSCP_SEC_RQ (または、PU 集信のケースではダウンストリーム LU に) X 定した場合には、 LU は、PLU-SLU セッションがこの期間非アクティブのまま D っていて以下の条件中の 1 つが保留中であるということが完了してから、非アクティブ化されます。

- セッションが限定 q 源リンクで渡される。
- セッションが再度 H 用される前に別のアプリケーションが LU を H 用したい。

タイムアウトがゼロにセットされている場合には、LU は非 h 動化されません。

**model_name**

パーソナル・コミュニケーションズまたは Communications Server はこのフィールドが EBCDIC 文 z の A-Z、0-9、@、# および $ から構成されているか検査します。このフィールドがすべて 2 進ゼロにセットされてはいなくて、ホスト・システムが SDDLU (Self-Defining Dependent LU;自己定義従属型 LU) をサポートしている場合、ホストにあるローカル LU を動的に定義するためにノードは非送信請求 PSID NMVT 応答を生成します。 PSID サブベクトルははこのフィールドに提供された名前が含まれています。

**name_attributes**

このビット・フィールドはシステムに提供された **base_name** の解釈とH用法を変更します。このフィールドはゼロの値、またはビット単位で互いに OR 結合された以下の値のいずれでも、またはすべてを取ることもできます。

**AP_USE_HEX_IN_NAME**

このビットが設定されると、**base_name** の解釈は次のように変更されます。

これは、6 バイトの英数 z タイプ A の EBCDIC 文 z 列 (先頭は文 z ）です。 EBCDIC スペースを右に埋めます。ベース名は、NAU 範囲内のF LU に対して 2 つの EBCDIC 文 z をU加され、NAU アドレスの16 進値を示します。

**AP_USE_BASE_NUMBER**

このビットが設定されると、**base_name** の解釈は次のように変更されます。

これは、5 バイトのタイプ A の EBCDIC ストリング（文 z で + O）による英数 z であり、右側には EBCDIC スペースを埋め込みます。このベース名は、3 つの EBCDIC 数値文 z をU加され、範囲内の LU の 10 進数X 8 を = し、**base_number** でOまり (**base_name + max_nau - min_nau**) で終わります。

**base_number**

AP_USE_BASE_NUMBER ビットが **name_attributes** にセットされない場合には、このフィールドは無 k されます。そうでない場合には、このフィールドは直前に説明された **base_name** の解釈を変更します。許可される値はゼロから (255 -**max_nau + min_nau**) までです。

**app_spec_def_data**

アプリケーションX定の定義データ。このフィールドはパーソナル・コミュニケーションズまたは Communications Server によって解釈されませんが、記憶された後 QUERY_LU_0_TO_3 verb によって戻されます（範囲内の LU ごとに同じデータが戻されます)。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_BASE_NUMBER

AP_INVALID_LU_MODEL
AP_INVALID_LU_NAME
AP_INVALID_NAME_ATTRIBUTES
AP_INVALID_NAU_ADDRESS
AP_INVALID_PRIORITY

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PU_NOT_DEFINED

AP_INVALID_PU_NAME
AP_INVALID_PU_TYPE
AP_LU_NAME_POOL_NAME_CLASH
AP_LU_ALREADY_DEFINED
AP_LU_NAU_ADDR_ALREADY_DEFD
AP_IMPLICIT_LU_DEFINED
AP_CANT_MODIFY_VISIBILITY

従属 LU サポートをH用してシステムが構築されていないために verb が実行されない場合、「プログラム」は次のパラメーターが戻されます。

**primary_rc**
AP_INVALID_VERB

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

## DEFINE_LU_0_TO_3_RANGE

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_LU_POOL

この verb をH用して LU プールを定義するか、LU を既存のプールに追加します。追加される LU は、DEFINE_LU_0_TO_3 verb または DEFINE_LU_0_TO_3_RANGE verb で定義されていなければなりません。 LU は、一度に 1 つの LU プールにだけ属します。X定した LU がすでに特定のプールに属している場合、その既存のプールから LU が除去され、定義するプールに追加されます。 1 つのプールへ一度に追加できる LU は 10 までですが、1 つのプール内の LU の合計数には限度はありません。

## VCB 構造体

```
typedef struct define_lu_pool
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   attributes;          /* verb attributes                */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned char   pool_name[8];        /* LU pool name                   */
        unsigned char   description[RD_LEN]; /* resource description           */
        unsigned char   reserv3[4];          /* reserved                       */
        unsigned short  num_lus;             /* number of LUs to add           */
        unsigned char   lu_names[10][8];     /* LU names                       */
} DEFINE_LU_POOL;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_LU_POOL

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**pool_name**

これらの LU が属するプールの名前。この名前は 8 バイトのストリングであり、右側にはスペースが埋め込まれます。さらにこの名前は、EBCDIC ストリングで = すこともできますし、ローカルに = 示可能な文 z セットによるストリングで = すこともできます。

**description**

q 源の説明（QUERY_LU_POOL で戻される)。このフィールドの長さは、4 バイトの倍数でなければならず、ゼロであってはなりません。

**num_lus**

追加する LU の数（範囲は 0 から 10)。

**lu_names**
プールに追加する LU の名前。それぞれの名前は、8 バイトのタイプ A の EBCDIC ストリング（文 z で + O）による英数 z であり、右側には EBCDIC スペースを埋め込みます。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_INVALID_NUM_LUS
AP_INVALID_POOL_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_LU_NAME_POOL_NAME_CLASH

AP_INVALID_POOL_NAME

従属 LU サポートをH用してシステムが構築されていないために verb が実行されない場合、「プログラム」は次のパラメーターが戻されます。

**primary_rc**
AP_INVALID_VERB

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_MODE

DEFINE_MODE verb は、一連のネットワーキング特性を定義し、特定のモード（またはセッションのグループ）にd り当てます。さらにこの verb をH用して、以前に定義したモードのフィールドがあればそれらを修正することもできます。 SNASVCMG モードを再定義する場合、その **mode_name** および **cos_name** は修正できません。CPSVCMG モードは再定義できません。

DEFINE_MODE verb をH用して、T明のモードがマップされるデフォルト COS を定義することもできます。これは、**mode_name** をすべてゼロに設定すると定義できます。デフォルト COS は、初めは #CONNECT です。

**注:** H用しているネットワーク・ノードおよび（可能ならば）パートナー・ノードでこれらのモードを定義する，要がありますが、ローカルにH用したいモードをすべて定義する，要はありません。定義されていないモードをX定するALLOCATE を発行すると、このノードは、DEFINE_DEFAULTS verb でX定したモデルとなるデフォルトのモードのためにその特性をH用します。そのようなモデルをX定していない場合、そのモデルにはブランクの特性のモードがH用されます。

## VCB 構造体

```
typedef struct define_mode
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned char   mode_name[8];        /* mode name                      */
        unsigned short  reserv3;             /* reserved                       */
        MODE_CHARS      mode_chars;          /* mode characteristics           */
} DEFINE_MODE;

typedef struct mode_chars
{
        unsigned char   description[RD_LEN]
                                             /* resource description           */
        unsigned short  max_ru_size_upp;     /* max RU size upper bound        */
        unsigned char   receive_pacing_win;  /* receive pacing window          */
        unsigned char   default_ru_size;     /* default RU size to maximize    */
                                             /* performance                    */
        unsigned short  max_neg_sess_lim;    /* max negotiable session limit   */
        unsigned short  plu_mode_session_limit;
                                             /* LU-mode session limit          */
        unsigned short  min_conwin_src;      /* min source contention winner   */
                                             /* sessions                       */
        unsigned char   cos_name[8];         /* class-of-service name          */
        unsigned char   cryptography;        /* cryptography                   */
        unsigned char   compression;         /* compression                    */
        unsigned short  auto_act;            /* initial auto-activation count  */
        unsigned short  min_conloser_src;    /* min source contention loser    */
        unsigned short  max_ru_size_low      /* maximum RU size lower bound    */
        unsigned short  max_receive_pacing_win;
                                             /* maximum receive pacing window  */
        unsigned char   max_compress_lvl;    /* maximum compression level      */
        unsigned char   max_decompression_lvl;
```

```
                                        /* maximum decompression level    */
        unsigned char   comp_in_series;  /* support for LZ and RLE         */
        unsigned char   reserv4[24];     /* reserved                       */
} MODE_CHARS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_MODE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドにゼロまたは 1 をセットしてください。

**mode_name**

モードの名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列 (先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。これがすべてゼロに設定されると、デフォルト COS は **mode_chars.cos_name** に設定され、他の **mode_chars** フィールドはすべて無ｋされます。

**mode_chars.compression**

このモードをHってｈ動化されたセッションに圧縮がH用されるかどうかを示します。

**AP_COMP_PROHIBITED**

このモードのセッションでは、RLE 圧縮はサポートされていません。

**AP_COMP_REQUESTED**

RLE 圧縮は、このモードのセッションでサポートされており要求されます（しかし、，須ではありません)。

**mode_chars.max_ru_size_upp**

このモードのセッションに送受信される RU の最大数の上限。セッションのｈ動化~に RU サイズの最大値を折衝によって決定する場合、この値がH用されます。範囲は 256 から 61440 までです。 **default_ru_size** が AP_YES に設定されていると、このフィールドは無ｋされます。

**mode_chars.receive_pacing_win**

このモードにおけるセッションのセッション・ペーシング・ウィンドウ。固定ペーシングの場合、この値は受信ペーシング・ウィンドウをX定します。適応ペーシングの場合、この値が好ましい最低限のウィンドウ・サイズとしてH用されます。隣接ノードが適応ペーシングをサポートしないとX定されない限り、パーソナル・コミュニケーションズまたは Communications Server は常に適応ペーシングをH用することに注意してください。値の範囲は 1〜63 です。ゼロの値は認められていません。

**mode_chars.default_ru_size**

RU サイズの最大値についてデフォルトの上限をH用するかどうかをX定します。このパラメーターで AP_YES をX定すると、**max_ru_size_upp** は無ｋされ、RU サイズの最大値の上限は、 TH と RH のサイズを引いたリンク BTU のサイズに設定されます。

AP_YES
AP_NO

**mode_chars.max_neg_sess_lim**
このモード上で任意のローカル LU とパートナー LU の間において許可されているセッションの最大数。ゼロの値をX定すると、暗黙の CNOS 交換は行われなくなります。値の範囲は、0〜32 767 です。

**mode_chars.plu_mode_session_limit**
このモードにおけるデフォルトのセッション限度。これは、このモード上で特定のローカル LU とパートナー LU の組み合わせにおけるセッションの数を制限します。CNOS（セッション数変更）の交換が暗黙に + O されるときに、この値がH用されます。ゼロの値をX定すると、暗黙の CNOS 交換は行われなくなります。値の範囲は、0〜32 767 です。

**mode_chars.min_conwin_src**
いずれかのローカル LU がこのモードをH用して h 動化できる競合勝者セッションの最小数。CNOS（セッション数変更）の交換が暗黙に + O されるときに、この値がH用されます。ゼロの値をX定すると、暗黙の CNOS 交換は行われなくなります。値の範囲は、0〜32 767 です。

**mode_chars.cos_name**
このモードでセッションを h 動化するときに要求するサービス・クラスの名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**mode_chars.cryptography**
セッション・レベルの暗号化をH用するかどうかをX定します（AP_NONE または AP_MANDATORY）。

**mode_chars.compression**
このモードをHって h 動化されたセッションに圧縮がH用されるかどうかを示します。

**AP_COMP_PROHIBITED**
このモードのセッションでは、圧縮はサポートされていません。

**AP_COMP_REQUESTED**
圧縮は、このモードのセッションでサポートされ、要求されます（しかし、 , 須ではありません）。

**format** フィールドが 0 にセットされている場合には、圧縮と解凍のレベルはノードがサポートする最大にセットされます。

**format** フィールドが 1 にセットされている場合には、圧縮と解凍の最大レベルが **max_compress_lvl** と **max_decompress_lvl** のフィールドにより定義されます。

**mode_chars.auto_act**
このモードで自動 h 動化させるセッションの数をX定します。セッション数変更 (CNOS) の交換が暗黙に + O されるときに、この値がH用されます。

範囲は、0 〜 32767 です。

**mode_chars.min_consloser_src**
このモードのどのローカル LU によっても h 動化されるためにコンテンショ

ン敗者セッションの最小値をX定します。CNOS（セッション数変更）の交換が暗黙に + O されるときに、この値がH用されます。範囲は、0 〜 32767 です。

**mode_chars.max_ru_size_low**

このモードのセッションに送受信される RU の最大サイズの下限をX定します。このセッションの h 動化 ~ に RU サイズの最大　体　**攅**の h

**mode_chars.comp_in_series**

RLE 圧縮が先行し、次に LZ 圧縮をというH用法が許可されるかどうかをX定します。このフィールドを AP_YES に設定すると、**max_compress_lvl** は AP_LZ9_COMPRESSION、AP_LZ10_COMPRESSION、または AP_LZ12_COMPRESSION にセットされなければなりません。

**AP_YES**

**AP_NO**

このフィールドは、ノードが (START_NODE の **comp_in_series** フィールドにX定された) RLE と LZ 圧縮をサポートしないように構成されている場合には、AP_YES にはセットできません。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_COS_NAME

AP_CPSVCMG_ALREADY_DEFD
AP_INVALID_CNOS_SLIM
AP_INVALID_COS_SNASVCMG_MODE
AP_INVALID_DEFAULT_RU_SIZE
AP_INVALID_MAX_NEGOT_SESS_LIM
AP_INVALID_MAX_RU_SIZE_UPPER
AP_INVALID_MAX_RU_SIZE_LOW
AP_RU_SIZE_LOW_UPPER_MISMATCH
AP_INVALID_COMPRESSION
AP_INVALID_MIN_CONWINNERS
AP_INVALID_MIN_CONLOSERS
AP_INVALID_MIN_CONTENTION_SUM
AP_INVALID_MODE_NAME
AP_INVALID_RECV_PACING_WINDOW
AP_INVALID_MAX_RECV_PACING_WIN
AP_INVALID_DEFAULT_RU_SIZES
AP_INVALID_SNASVCMG_MODE_LIMIT
AP_MODE_SESS_LIM_EXCEEDS_NEG
AP_INVALID_CRYPTOGRAPHY

AP_INVALID_MAX_COMPRESS_LVL
AP_INVALID_MAX_DECOMPRESS_LVL
AP_INVALID_COMP_IN_SERIES

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

**再定 A の / 効**

F フィールドの再定義の結果は以下のとおりです。

**description**
更新済み **description** は次回の QUERY_MODE verb で戻されます。

**圧 L**

**max_compress_lvl**

**max_decompress_lvl**

**comp_in_series**

**暗号化**

**max_ru_size_upp**

**receive_pacing_win**

**default_ru_size**

**max_ru_size_low**

**max_receive_pacing_win**
更新済みの値はこのモードのすべての後続セッションの h 動化 n 行に H 用され、すべての後続 QUERY_MODE verb で戻されます。変更はどの既存アクティブ・セッションにも効きません。

**max_neg_sess_lim**

**plu_mode_session_limit**

**min_conwin_src**

**auto_act**

**min_conloser_src**
更新済み値は、次の CNOS コマンド (ローカル側での + O またはリモート側

での + O いずれでも) までは特定のローカル LU またはパートナー LU のためにはH用されません。古い値は次の CNOS コマンドまでは QUERY_MODE verb に入れて戻されます。

**cos_name**

更新済み値はこのモードのすべての次回セッションh動化n行にH用され、すべての次回 QUERY_MODE verb で戻されます。変更はどの既存アクティブ・セッションにも効きません。更新済み値は COS マッピング操作へのどんな後続のモード (たとえば、このノードがネットワーク・ノードであり、COS マッピング・サービスまたはそのサーブされるエンド・ノードへモードを提供する場合) にもH用され、すべての後続 QUERY_MODE_TO_COS_MAPPING verb で戻されます。

**注:** 暗黙のモード定義は DEFINE_MODE によって ’明示的’ に変えられます。これは、**implicit set** を AP_NO に戻す後続の QUERY_MODE verb により反映されます。

# DEFINE_PARTNER_LU

DEFINE_PARTNER_LU verb は、ローカル LU とパートナー LU との間の LU-LU セッションについて、パートナー LU のパラメーターを定義します。もしくは、DEFINE_PARTNER_LU をH用して、**fqplu_name** および **plu_alias** 以 0 のパートナー LU にすでに定義されているすべてのパラメーターを修正することができます。

## VCB 構造体

```
typedef struct define_partner_lu
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   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[RD_LEN]; /* resource description           */
        unsigned char   plu_un_name[8];      /* partner LU uninterpreted name  */
        unsigned char   preference           /* routing preference             */
        unsigned short  max_mc_ll_send_size; /* max MC send LL size            */
        unsigned char   conv_security_ver;   /* already_verified accepted?     */
        unsigned char   parallel_sess_supp;  /* parallel sessions supported?   */
        unsigned char   reserv2[8];          /* reserved                       */
} PLU_CHARS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_PARTNER_LU

**format**

VCB の形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**plu_chars.fqplu_name**

パートナー LU の完全修飾名。この名前の長さは 17 バイトで、 EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文z の文z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

**plu_chars.plu_alias**

パートナー LU のエイリアス。これは、ローカル=示可能文z セットの 8 バイトの文z 列です。関連Uけられたエイリアスがないパートナー LU の場合には、このフィールドはすべてゼロに設定できます。

**plu_chars.description**

q源の説明（QUERY_PARTNER_LU および

QUERY_PARTNER_LU_DEFINITION で戻される)。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**plu_chars.plu_un_name**
パートナー LU の非解釈名。これは、8 バイトのタイプ A EBCDIC 文 z ストリングです。

**plu_chars.max_mc_ll_send_size**
パートナー LU においてマップ式会話サービスによって送受信される LL レコードの最大数。値の範囲は 1-32 767 です。(32 767 は、このフィールドを 0 に設定することによりフィールド X 定します。)

**plu_chars.preference**
このパートナー LU へのセッション h 動化のために H 用するときの、望ましい経路 X 定プロトコル。このフィールドは、以下の値をとることができます。

**AP_NATIVE**
ネイティブ (APPN) 経路 X 定プロトコルのみを H 用します。

**AP_NONNATIVE**
非ネイティブ (AnyNet) プロトコルのみを H 用します。

**AP_NATIVE_THEN_NONNATIVE**
ネイティブの (APPN) プロトコルを n 行し、パートナー LU が見つからなければ、非ネイティブの (AnyNet) プロトコルを H ってセッション h 動化を再 n 行します。

**AP_NONNATIVE_THEN_NATIVE**
非ネイティブ (AnyNet) プロトコルを n 行し、パートナー LU を見つけることができない場合には、ネイティブ (APPN) プロトコルを H 用してセッション h 動化を再 n 行します。

**AP_USE_DEFAULT_PREFERENCE**
ノードの + O ~ に定義したデフォルトのプリファレンスを H 用する。(これは QUERY_NODE によって再び呼び出されます。)

**注:** 非ネイティブ経路 X 定は、AnyNet DLC がノード・オペレーター機能で利用可能であり、定義済みの AnyNet リンク・ステーションがある場合にのみ意味をもちます。(Defined_LS を 2 照)。

**plu_chars.conv_security_ver**
ローカル LU に代わってパートナー LU で **user_ids** の妥当性検査が行われることを許可するかどうか、つまりパートナー LU はすでに検査された X 示を Attach 要求で設定できるかどうかを X 定します (AP_YES または AP_NO)。

**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 X 定します (AP_YES または AP_NO)。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_ANYNET_NOT_SUPPORTED

AP_DEF_PLU_INVALID_FQ_NAME
AP_INVALID_UNINT_PLU_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PLU_ALIAS_CANT_BE_CHANGED

AP_PLU_ALIAS_ALREADY_USED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

**再定 A の / 効**

F フィールドの再定義の結果は以下のとおりです。

**fqplu_name**
変更できません。

**plu_alias**

直前の DEFINE_PARTNER_LU が 別の **plu_alias** で発行された場合には、DEFINE_PARTNER_LU は失敗します。直前の DEFINE_PARTNER_LU がすべてゼロの **plu_alias** で発行された場合には、再定義は受け入れられて、すべての既存 PLU レコードを有効にします。直前の DEFINE_PARTNER_LU が発行されなかった場合には、すべてゼロがX定されない限り、X定された **plu_alias** はすべての対応した暗黙的に定義されたパートナー LU レコードにコピーされます。すべてゼロがX定されれば暗黙的 **plu_aliases** はT変のままです。

**注:** アプリケーションがすでに暗黙的 **plu_alias** を前の APPC verb から入手して、それを後続の ALLOCATE でH用している場合には、非ゼロの **plu_alias** で DEFINE_PARTNER_LU を発行すると実行中のアプリケーションに障 2 が起こることがありえます。

**description**

更新済み **description** は後続の QUERY_PARTNER_LU verb で戻されます。

**plu_un_name**

更新済み **plu_un_name** は、このパートナー LU へのすべての後続セッションh動化要求にH用され、すべての後続 QUERY_PARTNER_LU verb で戻されます。

**preference**

更新済み **preference** は、このパートナー LU へのすべての後続セッションh動化要求にH用され、すべての後続 QUERY_PARTNER_LU verb で戻されます。

**max_mc_ll_send_size**

更新済み **preference** は、このパートナー LU (既存セッションのものでも)へのすべての後続セッションh動化要求にH用されます。変更は既存の会話には効きません。更新済みの値はすべての後続 QUERY_PARTNER_LU verb で戻されます。

**conv_security_ver**

更新済み値は、そのローカル LU とパートナー LU 間のセッションの数がゼロになるまでは、特定のローカル LU のためにはH用されません。 BIND と RSP (BIND) は古い設定をH用して流れ、また古い値はセッションの数がゼロになるまで QUERY_PARTNER_LU で戻されます。これは、機密保護サポートが既存のアクティブ・セッションのものとは異なる場合、パートナー LU は後続のセッションh動化n行をリジェクトできないためです。

**parallel_sess_supp**

**conv_security_ver** の場合と同様に、更新済みの値は、そのローカル LU とX定されたパートナー LU 間のセッションの数がゼロになるまでは、特定のローカル LU のためにはH用されません。これは構築された LU6.2 セッションの整合性検査の問題を避けるためです。

**注:** 暗黙のモード定義は DEFINE_PARTNER_LU によって ’明示的’ に変えられます。これは、AP_NO に**暗黙のセット** を戻す後続の QUERY_PARTNER_LU verb により反映されます。

# DEFINE_PORT

DEFINE_PORT は、新しいポートを定義するか、既存のポートを修正します。このポートはX定した DLC に属しますが、この DLC は DEFINE_DLC verb をH用して定義されていなければなりません。DEFINE_PORT verb は、動的リンク・ステーションでのH用のために、ポート特有のパラメーターおよびデフォルトの LS 特性をX定するだけではなく、ノード全体で固有なポート名をUけます。ポート特有のパラメーターは、基本構造体に連結されます。デフォルトの LS 特性は、ポート特有のパラメーターのすぐ後で連結されます。

（STOP_PORT の発行後に）ポートがリセット状態であり、かつ以前にこのポートを定義した~以降、DEFINE_PORT でX定された **dlc_name** が変更を加えられていない場合には、 DEFINE_PORT をH用して既存のポート上の 1 つ以上のフィールドを修正できます。

ポートがアクティブの場合には、以下のフィールドだけが変更可能です。

description
implicit_dspu_services
implicit_deact_timer
implicit_cp_cp_sess_support
implicit_link_lvl_error
default_tg_chars
implicit_dspu_template
implicit_ls_limit
link_spec_data_len
link_spec_data

ポートがアクティブの間にポートE様のデータが変更された場合には、verb はリジェクトされませんが、変更は無 k されます。

DLC、ポート、およびリンク・ステーションの関連についての詳細は、15ページの『DLC プロセス、ポート、リンク・ステーション』を2照してください。

## VCB 構造体

```
typedef struct define_port
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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;         /* resource description          */
        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 links to EN are      */
```

```
                                        /* uplink                          */
        unsigned char   reserv3[2];     /* reserved                        */
        unsigned long   port_number;    /* port number                     */
        unsigned short  max_rcv_btu_size;  /* max receive BTU size         */
        unsigned short  tot_link_act_lim;  /* total link activation limit  */
        unsigned short  inb_link_act_lim;  /* inbound link activation limit*/
        unsigned short  out_link_act_lim;  /* outbound link activation     */
                                        /* limit                           */
        unsigned char   ls_role;        /* initial link station role       */
        unsigned char   retry_flags;    /* conditions for automatic        */
                                        /* retries                         */
        usigned char    max_activation_attempts;
                                        /* how many automatic retries?     */
        unsigned char   adtivation_delay_timer;
                                        /* delay between automatic         */
                                        /* retries                         */
        unsigned char   reserv1[10];    /* reserved                        */
        unsigned char   implicit_dspu_template[8];
                                        /* reserved                        */
        unsigned char   implicit_ls_limit;  /* max number of implicit links*/
        unsigned char   reserv2;        /* reserved                        */
        unsigned char   implicit_dspu_services;
                                        /* implicit links support DSPUs    */
        unsigned char   implicit_deact_timer;
                                        /* Implicit link HPR link          */
                                        /* deactivation timer              */
        unsigned short  act_xid_exchange_limit;
                                        /* act.  XID exchange limit        */
        unsigned short  nonact_xid_exchange_limit;
                                        /* nonact.  XID exchange limit     */
        unsigned char   ls_xmit_rcv_cap;    /* LS transmit-receive         */
                                        /* capability                      */
        unsigned char   max_ifrm_rcvd;  /* max number of I-frames that     */
                                        /* can be received                 */
        unsigned short  target_pacing_count; /* Target pacing count        */
        unsigned short  max_send_btu_size;  /* Desired max send BTU size   */
        LINK_ADDRESS    dlc_data;       /* DLC data                        */
        LINK_ADDRESS    hpr_dlc_data;   /* HPR DLC data                    */
        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;
                                        /* Implicit links support HPR      */
                                        /* link-level error recovery       */
        unsigned char   retired1;       /* reserved                        */
        TG_DEFINED_CHARS default_tg_chars;  /* Default TG chars            */
        unsigned char   discovery_support   /* Discovery function          */
                                        /* supported?                      */
        unsigned short  port_spec_data_len; /* length of port spec data    */
        unsigned short  link_spec_data_len; /* length of link spec data    */
} PORT_DEF_DATA;

typedef struct link_address
{
        unsigned short  length;         /* length                          */
        unsigned short  reserve1;       /* reserved                        */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                        /* address                         */
} LINK_ADDRESS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_PORT

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**port_name**

定義するポートの名前。これは、ローカル=示可能文zセットの 8 バイトの文z列です。8 バイトすべてが有効であり、すべて設定する,要があります。

**def_data.description**

q源の説明（QUERY_PORT で戻される)。これは、ローカルに=示可能な文zセットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**def_data.dlc_name**

関連Uけられた DLC の名前であり、ローカルに=示可能な文zセットによる 8 バイトのストリングです。8 バイトすべてが有効であり、すべて設定する,要があります。名前のついた DLC は、DEFINE_DLC verb によって定義されている,要があります。

**def_data.port_type**

ポートがH用する回線のタイプをX定します。この値は、以下の回線タイプの 1 つに対応しています。

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

このフィールドを AP_PORT_SATF に設定する場合、**ls_role** を AP_LS_NEG に設定しなければならないことに注意してください。

**def_data.port_attributes[0]**

これはビット・フィールドです。それは値 AP_NO をとるか、または以下の値を取ります。

**AP_RESOLVE_BY_LINK_ADDRESS**

これは、着呼を解決するために受信された XID3 で運ばれた CP 名( または、ノード ID) をH用する前に、CONNECT_IN のリンク・アドレスをH用することにより着呼の解決がn行されることをX定します。このビットは、フィールド **port_type** が AP_PORT_SWITCHED にセットされていなければ、無kされます。

**def_data.implicit_uplink_to_en**

BrNN のみ: 隣接ノードがエンド・ノードである場合に、このポートをオフにしている暗黙的リンク・ステーションがアップリンクであるかダウンリンクであるかをX定します。このフィールドの値は、同一パートナーへの既存リンクがない場合にのみ考慮されます。そのようなリンクはリンク・タイプを判別するために最初にH用されるからです。

**AP_NO**

暗黙のリンクはダウンリンクです。

**AP_YES**

暗黙のリンクはアップリンクです。

他のノード・タイプ : このフィールドは無 k されます。

**def_data.port_number**

ポート番号。

**def_data.tot_link_act_lim**

合計リンク h 動化限度。ここでは、同 ~ に h 動化できるリンク・ステーションの最大数をX定します。これは、**inb_link_act_lim** フィールドと **out_link_act_lim** フィールドの合計以上でなければなりません。 **port_type** を AP_PORT_NONSWITCHED に設定し、 **ls_role** を AP_LS_NEG または AP_LS_SEC に設定する場合、このフィールドは 1 に設定する , 要があります。**ls_role** を AP_LS_PRI に設定する場合、このフィールドの値の範囲は 1 から 256 である , 要があります。このポートが AnyNet DLC 用である場合、65535 をH用しなければなりません。

**def_data.inb_link_act_lim**

インバウンド・リンク h 動化限&。ここでは、このポート上でのインバウンド h 動化のために予約されているリンク・ステーションの数をX定します。同 ~ に h 動化できるアウトバウンド・リンク・ステーションの最大数は、**def_data.tot_link_act_lim** から **def_data.inb_link_act_lim** の範囲になります。**port_type** を AP_PORT_NONSWITCHED に設定し、 **ls_role** を AP_LS_NEG または AP_LS_PRI に設定する場合、このフィールドは 0 に設定する , 要があります。**port_type** を AP_PORT_NONSWITCHED に設定し **ls_role** を AP_LS_SEC に設定する場合、このフィールドは 0 か 1 に設定しなければなりません。このポートが AnyNet DLC 用である場合、0 をH用しなければなりません。

**def_data.out_link_act_lim**

アウトバウンド・リンク h 動化限&。ここでは、このポート上でのアウトバウンド h 動化のために予約されているリンク・ステーションの数をX定します。同 ~ に h 動化できるインバウンド・リンク・ステーションの最大数は、**def_data.tot_link_act_lim** から **def_data.out_link_act_lim** の範囲になります。 **port_type** を AP_PORT_NONSWITCHED に設定し、**ls_role** を AP_LS_NEG に設定する場合、このフィールドは 0 に設定する , 要があります。**ls_role** を
AP_LS_PRI に設定する場合、このフィールドの値は
**tot_link_act_lim** と同じ値にしなければなりません。 **port_type** を AP_PORT_NONSWITCHED に設定し **ls_role** を AP_LS_SEC に設定する場合、このフィールドは 0 か 1 に設定しなければなりません。このポートが AnyNet DLC 用である場合、0 をH用しなければなりません。

**def_data.ls_role**

リンク・ステーションのロール。これは交渉可能 (AP_LS_NEG)、1 次 (AP_LS_PRI)、または 2 次 (AP_LS_SEC) のいずれかになります。このリンク・ステーションのロールにより、前述の **tot_act_lim**、 **inb_link_act_lim**、および **out_link_act_lim** フィールドによってX定した値の間の関連を判別します。**port_type** を AP_PORT_SATF に設定する場合、 **ls_role** を AP_LS_NEG に設定しなければならないことに注意してください。

**def_data.retry_flags**

このフィールドは、フラグ AP_INHERIT_RETRY が **def_data.retry_flags** の DEFINE_LS にセットされる場合に、このリンク・ステーションの h 動化が自動的再 n 行の対象となる条件をX定します。それはビット・フィールドで、以下の値をビット単位で互いに OR 結合したいずれのものでも可能です。

**AP_RETRY_ON_START**

h 動化が n 行されたときにリモート・ノードから無応答の場合には、リンク h 動化が再 n 行されます。 h 動化が n 行されたときに基になっているポートが非 h 動である場合には、APPN はそれを h 動化しようとします。

**AP_RETRY_ON_FAILURE**

リンクが h 動中、または h 動保留 ~ に障 2 があれば、リンク h 動化が再 n 行されます。 h 動化が n 行されたときに基になっているポートに障 2 があれば、APPN はそれを h 動化しようとします。

**AP_RETRY_ON_DISCONNECT**

リンクがリモート・ノードにより正規に停 _ されれば、リンク h 動化が再 n 行されます。

**AP_DELAY_APPLICATION_RETRIES**

リンク h 動化は、アプリケーション (START_LS またはオンデマンド・リンク h 動化を H 用したもの) により + O されて、**activation_delay_timer** をH用してペースを取られます。

**AP_INHERIT_RETRY**

このフィールドのフラグがX定する再 n 行条件に加えて、基になっているポート定義の **retry_flags** フィールドでX定したこれらのものもH用されます。

**def_data.max_activation_attempts**

このフィールドは、少なくとも 1 つのフラグが **def_data.retry_flags** の DEFINE_LS 内にセットされ、 DEFINE_LS の **def_data.max_activation_attempts** が AP_USE_DEFAULTS にセットされていなければ、効力がありません。

このフィールドは、リモート・ノードが応答しないか、または基になっているポートが非 h 動であるときに、「プログラム」 が許可する再 n 行の数をX定します。これには自動的再 n 行とアプリケーション主導型の h 動化 n 行が含まれます。

この限&に達すると、それ以上の自動的再 n 行は行われません。この条件は STOP_LS、STOP_PORT、STOP_DLC または h 動化の成功によりリセットされます。 START_LS または OPEN_LU_SSCP_SEC_RQ は 1 回 だけの h 動化 n 行に終わります、 h 動化が失敗しても再 n 行は行われません。

ゼロは ’限 & がない’ ことを意味します。値 AP_USE_DEFAULTS は DEFINE_DLC でシステムに提供された **max_activiation_attempts** のH用という結果になります。

**def_data.activation_delay_timer**
このフィールドは、少なくとも

す（さらに、**def_data.implicit_dspu_template** フィールドでX定された DSPU テンプレートのような定義を書き込みます）。

**AP_NONE**
ローカル・ノードは、このダウンストリーム PU にサービスを提供しません。

**def_data.implicit_deact_timer**
限度のある q 源のリンク非 h 動化タイマー（C 数）。**implicit_limited_resource** を AP_YES または AP_NO_SESSIONS に設定する場合で、このタイマーで設定した~間内に HPR 機能がある暗黙のリンクを走査するデータがなく、このリンクをH用しているセッションがない場合は、このリンクは自動的に非 h 動化されます。

**implicit_limited_resource** が AP_INACTIVITY に設定されると、このタイマーの設定~間内にリンクを流れるデータがない場合、暗黙リンクは自動的に非 h 動化されます。

値は 0-1000 Cの範囲の整数です。デフォルトは 10 Cです。

ゼロをX定する場合、デフォルト値として 30 CがH用されます。それ以 0 の場合、最小値は 5 です。（これより小さい値をX定しても、X定した値は無 k され、5 がH用されます。）**implicit_limited_resource** が AP_NO に設定されない限り、このパラメーターは予約済みになることに留意してください。

**def_data.act_xid_exchange_limit**
h 動化 XID の交換限度。

**def_data.nonact_xid_exchange_limit**
非 h 動化 XID の交換限度。

**def_data.ls_xmit_rcv_cap**
リンク・ステーションの伝送／受信機能を設定します。この機能は、両方向同~通信 (AP_LS_TWS)（二重方式または全二重方式ともいう）、または両方向交互通信 (AP_LS_TWA)（半二重方式ともいう）のいずれかです。

**def_data.max_ifrm_rcvd**
肯定応答の送信前にローカル・リンク・ステーションによって受信できる I フレームの最大数。この値の範囲は 1〜127 です。

**def_data.target_pacing_count**
1 から 32 767 の数値であり、この TG での BIND 用の望ましいペーシング・ウィンドウ・サイズを包 g 的に示します。この番号は、固定バインド・ペーシングの実行~にのみ有効です。パーソナル・コミュニケーションズまたは Communications Server は、現在はこの値をH用しないことに注意してください。

**def_data.max_send_btu_size**
このリンク・ステーションから送信できる BTU サイズの最大値。この値をH用して、1 組のリンク・ステーション間で伝送できる BTU サイズの最大値を決定します。HPR 機能がある暗黙のリンクがそのポート上でサポートされていない場合、この値は 99 以上に設定する，要があります。 HPR 機能がある暗黙のリンクがそのポート上でサポートされている場合、この値は 768 以上に設定する，要があります。

**def_data.dlc_data.length**
ポート・アドレスの長さ。

**def_data.dlc_data.address**
ポート・アドレス。

**def_data.hpr_dlc_data.length**
HPR ポート・アドレスの長さ。

**def_data.hpr_dlc_data.address**
HPR ポート・アドレス。これは現在、HPR リンクをサポートしているときにH用されます。このフィールドは、このポートを介してリンク・ステーション上で交換される XID3 に含まれる X'61' の制御ベクトルの X'80' のサブフィールドによって、パーソナル・コミュニケーションズまたは Communications Server が送信する情報をX定します。パーソナル・コミュニケーションズまたは Communications Server が DLC に発行した ACTIVATE_PORT をH用して渡されます。DLC によっては、HPR リンクをサポートするポートのために、この情報が , 要であるものもあります。

**def_data.implicit_cp_cp_sess_support**
このポートをオフにしている暗黙のリンク・ステーションに CP-CP セッションを許可するかどうかをX定します（AP_YES または AP_NO）。

**def_data.implicit_limited_resource**
リンクをH用しているセッションがない場合に、このポートをオフにしている暗黙のリンク・ステーションを非 h 動化するかどうかをX定します。これは、以下の値の 1 つに設定されます。

**AP_NO**
暗黙のリンクは限定された q 源ではなく、自動的に非 h 動化されることはありません。

**AP_YES または AP_NO_SESSIONS**
暗黙のリンクは限定された q 源であり、リンクをH用している h 動セッションがない場合に、自動的に非 h 動化されます。

**AP_INACTIVITY**
暗黙リンクは限定 q 源であり、それらのリンクをH用しているアクティブ・セッションがない場合、または **implicit_deact_timer** フィールドでX定した ~ 間内にデータがリンク上を流れない場合、自動的に非 h 動化されます。

**def_data.implicit_hpr_support**
暗黙のリンク上で HPR をサポートするかどうかをX定します（AP_YES または AP_NO）。

**def_data.implicit_link_lvl_error**
リンク・レベルのエラー回 | をH用して暗黙のリンク上で HPR 通信を送信するかどうかをX定します（AP_YES または AP_NO）。**implicit_hpr_support** を AP_NO に設定すると、このパラメーターは予約されることに注意してください。

**def_data.default_tg_chars**
TG 特性（37ページの『DEFINE_COS』を 2 照）。これは、このポートをオフ

にしている暗黙のリンク・ステーションおよび **use_default_tg_chars** をX定する定義済みのリンク・ステーション向けにH用されます。

**def_data.discovery_supported**
このポート上で「検出」機能を実行するかどうかをX定します（AP_YES または AP_NO）。

**def_data.port_spec_data_len**
ACTIVATE_PORT 信号によって未変更のままポートに渡されるデータの長さ。このデータは、基本構造体に連結する，要があります。

**def_data.link_spec_data_len**
このフィールドは、常にゼロに設定されていなければなりません。

## 戻りＱｉａー？ー

verb の処理が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PORT_NAME

AP_INVALID_DLC_NAME
AP_INVALID_PORT_TYPE
AP_INVALID_BTU_SIZE
AP_INVALID_LS_ROLE
AP_INVALID_LINK_ACTIVE_LIMIT
AP_INVALID_MAX_IFRM_RCVD
AP_INVALID_DSPU_SERVICES
AP_HPR_NOT_SUPPORTED
AP_DLUR_NOT_SUPPORTED
AP_PU_CONC_NOT_SUPPORTED
AP_INVALID_TEMPLATE_NAME
AP_INVALID_RETRY_FLAGS
AP_INVALID_IMPLICIT_UPLINK

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PORT_ACTIVE

AP_DUPLICATE_PORT_NUMBER
AP_CANT_MODIFY_WHEN_ACTIVE
AP_CANT_MODIFY_VISIBILITY
AP_INVALID_IMPLICIT_UPLINK

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_TP

DEFINE_TP verb は、ノード・オペレーター機能トランザクション・プログラム (TP) 接続マネージャーがパートナー LU からの着信接続を処理するときにH用する TP 情報を定義します。さらにこの verb をH用して、以前に定義したトランザクション・プログラムのフィールドを 1 つ以上修正することもできます（ただし、パーソナル・コミュニケーションズまたは Communications Server が定義したトランザクション・プログラムを修正するためにはH用できません）。

## VCB 構造体

```
typedef struct define_tp
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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[RD_LEN]  /* resource description          */
        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   duplex_support;      /* duplex supported              */
        unsigned char   reserv3[9];          /* reserved                      */
        unsigned short  tp_instance_limit;   /* limit on currently active TP  */
                                             /* instances                     */
        unsigned short  incoming_alloc_timeout;
                                             /* incoming allocation timeout   */
        unsigned short  rcv_alloc_timeout;   /* receive allocation timeout    */
        unsigned short  tp_data_len;         /* TP data length                */
        TP_SPEC_DATA    tp_data;             /* TP data                       */
} TP_CHARS;

typedef struct tp_spec_data
{
        unsigned char pathname[256];         /* path and TP name              */
        unsigned char parameters[64];        /* parameters for TP             */
        unsigned char   queued;              /* queued TP                     */
        unsigned char load_type;             /* type of load-DETACHED/CONSOLE */
        unsigned char   dynamic_load         /* dynamic loading of TP enabled */
        unsigned char   reserved[5];         /* reserved                      */
} TP_SPEC_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_TP

**attributes**
verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**tp_name**
定義するトランザクション・プログラム (TP) の名前。これは、64 バイトの EBCDIC ストリングであり、右側には EBCDIC スペースを埋め込みます。パーソナル・コミュニケーションズまたは Communications Server はこのフィールドの文 z セットを検査しないことに注意してください。

**tp_chars.description**
q 源の説明（QUERY_TP_DEFINITION および QUERY_TP で戻される）。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**tp_chars.conv_type**
このトランザクション・プログラムでサポートする会話のタイプを X 定します。

AP_BASIC
AP_MAPPED
AP_EITHER

**tp_chars.security_rqd**
トランザクション・プログラムを + O するために会話機密保護情報が , 要かどうかを示します（AP_NO または AP_YES）。

**tp_chars.sync_level**
このトランザクション・プログラムでサポートする同期レベルを X 定します。

**AP_NONE**
トランザクション・プログラムは None の同期レベルをサポートします。

**AP_CONFIRM_SYNC_LEVEL**
トランザクション・プログラムは Confirm の同期レベルをサポートします。

**AP_EITHER**
トランザクション・プログラムは None または Confirm の同期レベルをサポートします。

**AP_SYNCPT_REQUIRED**
トランザクション・プログラムは Sync-point の同期レベルをサポートします。

**AP_SYNCPT_NEGOTIABLE**
トランザクション・プログラムは None、Confirm または Sync-point の同期レベルをサポートします。

**tp_chars.dynamic_load**
トランザクション・プログラムを動的にロードできるかどうかをX定します（AP_YES または AP_NO）。

**tp_chars.enabled**
トランザクション・プログラムを正常に接続できるかどうかをX定します（AP_YES または AP_NO）。デフォルトは AP_NO です。

**tp_chars.pip_allowed**
トランザクション・プログラムがプログラム初期設定 (PIP) パラメーターを受信できるかどうかを示します（AP_YES または AP_NO）。

**tp_chars.duplex_support**
トランザクション・プログラムが全二重方式であるのか半二重方式であるのかを示します。

**AP_FULL_DUPLEX**
トランザクション・プログラムが全二重方式であることをX定します。

**AP_HALF_DUPLEX**
トランザクション・プログラムが半二重方式であることをX定します。

**AP_EITHER_DUPLEX**
トランザクション・プログラムを半二重方式と全二重方式のどちらにも設定できることをX定します。

**tp_chars.tp_instance_limit**
同 ~ に h 動化するトランザクション・プログラムのインスタンスの限度数。ゼロという値は限度がないことを意味しています。

**tp_chars.incoming_alloc_timeout**
着信接続が RECEIVE_ALLOCATE を待機して待ち行列化されるときのC数をX定します。ゼロの値はタイムアウトがないことを示すので、無期限の保留状態になります。

**tp_chars.rcv_alloc_timeout**
接続を待機する間に RECEIVE_ALLOCATE verb が待ち行列化されているC数をX定します。ゼロの値はタイムアウトがないことを示すので、無期限の保留状態になります。

**tp_chars.tp_data_len**
システム依存のトランザクション・プログラム・データの長さ。

**tp_spec_data**
トランザクション・プログラムをランチするときに接続マネージャーがH用する情報。接続マネージャーのH用方法の詳細は、*パーソナル・コミュニケーションズ・クライアント/サーバー・コミュニケーション・プログラミング*にある「接続マネージャー」を 2 照してください。

**tp_chars.tp_data.pathname**
パスおよびトランザクション・プログラム名をX定します。

**tp_chars.tp_data.parameters**
トランザクション・プログラムのパラメーターをX定します。

**tp_chars.tp_data.queued**
トランザクション・プログラムを待ち行列化するかどうかをX定します。

**tp_chars.tp_data.load_type**
トランザクション・プログラムをロードする方法をX定します。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_SYSTEM_TP_CANT_BE_CHANGED

AP_INVALID_CONV_TYPE
AP_INVALID_SYNC_LEVEL
AP_INVALID_DYNAMIC_LOAD
AP_INVALID_ENABLED
AP_INVALID_PIP_ALLOWED
AP_INVALID_DUPLEX_SUPPORT

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_CANT_MODIFY_VISIBILITY

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

**再定Aの / 効**: F フィールドの再定義は即~有効になります (たとえば、トランザクション・プログラムの次のインスタンスが + O したとき)。しかし、フィールド **incoming_alloc_timeout** と **rcv_alloc_timeout** への変更は既にキューに入っているどの「接続」も RECEIVE_ALLOCATES も有効にしません。

# DELETE_ADJACENT_NODE

DELETE_ADJACENT_NODE は、隣接ノード上の q 源と関連U けられたノード・ディレクトリー・データベースにある項目を除去します。

ノードの制御点をその LU と共にディレクトリーから除去するには、**num_of_lus** をゼロに設定します。**num_of_lus** がゼロ以 0 である場合、この verb をH用してノード LU をディレクトリーから除去しますが、制御点の定義はそのままになります。

この verb が何らかの理由で失敗すると、ディレクトリー項目は削除されません。

## VCB 構造体

DELETE_ADJACENT_NODE verb には、ADJACENT_NODE_LU オーバーレーの変数番号が含まれています。ADJACENT_NODE_LU 構造体は、
DELETE_ADJACENT_NODE 構造体の終わりに連結されます。

```
typedef struct delete_adjacent_node
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   cp_name[17];        /* CP name                       */
        unsigned short  num_of_lus;         /* number of LUs                 */
} DELETE_ADJACENT_NODE;

typedef struct adjacent_node_lu
{
        unsigned char   wildcard_lu;        /* wildcard LU name indicator    */
        unsigned char   fqlu_name[17];      /* fully qualified LU name       */
        unsigned char   reserv1[6];         /* reserved                      */
} ADJACENT_NODE_LU;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_ADJACENT_NODE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**cp_name**

隣接 LEN エンド・ノードにおける制御点の完全修飾名。この名前の長さは 17 バイトであり、EBCDIC スペースが右に埋め込まれています。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

**num_of_lus**

削除する LU の数。ノード定義全体を削除するのであれば、このフィールドをゼロに設定してください。この数は、DELETE_ADJACENT_NODE VCB に続く隣接 LU オーバーレーの数を示します。

**adjacent_node_lu.wildcard_lu**
X定した LU 名がワイルドカード名（AP_YES または AP_NO）であるかどうかを示します。

**adjacent_node_lu.fqlu_name**
削除する LU 名。この名前が完全修飾されていない場合、CP 名のネットワーク ID であると見なされます。この名前の長さは 17 バイトであり、EBCDIC スペースが右に埋め込まれています。 この名前は 1 つか 2 つのタイプ A の EBCDIC 文 z ストリングで構成され、これらは EBCDIC ドットで連結されています。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_CP_NAME

AP_INVALID_LU_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_CP_NAME

AP_INVALID_LU_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**DELETE_ADJACENT_NODE**

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_CN

DELETE_CN は、関連Uけられているポートすべてをリセットする場合に、接続ネットワーク制御ブロックのメモリーを削除することによりそのメモリーを解放します。さらに、DELETE_CN をH用して、選択したポートを接続ネットワークから削除することもできます。そのためには、**num_ports** フィールドをゼロ以 0 の値に設定し、削除するポートのポート名を入力する , 要があります。

## VCB 構造体

```
typedef struct delete_cn
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   fqcn_name[17];       /* name of connection network    */
        unsigned char   reserv1;             /* reserved                      */
        unsigned short  num_ports;           /* number of ports to delete     */
        unsigned char   port_name[8] [8];    /* names of ports to delete      */
} DELETE_CN;
```

## 指定Ｑｉａ－？－

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_CN

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**fqcn_name**

削除する接続ネットワークの名前（17 バイト長)。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式でX定します。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

**num_ports**

接続ネットワーク上で削除するポートの数。接続ネットワーク全体を削除する場合には、このフィールドをゼロに設定しなければなりません。

**port_name**

**num_ports** がゼロ以 0 の値である場合に、削除するポートの名前。 F ポート名は、ローカルに = 示可能な文 z セットによる 8 バイトのストリングです。8 バイトすべてが有効であり、すべて設定する , 要があります。 **num_ports** フィールドをゼロに設定すると、このフィールドは予約されます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_CN_NAME

AP_INVALID_NUM_PORTS_SPECIFIED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_COS

DELETE_COS は、サービス・クラス項目が SNA によって定義されたサービスのデフォルト・クラスの 1 つでなければ、そのサービス・クラス項目を削除します。

## VCB 構造体

```
typedef struct delete_cos
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   reserv2;            /* reserved                     */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   cos_name[8];        /* class-of-service name        */
} DELETE_COS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_DELETE_COS

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**cos_name**
サービス・クラス名。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_COS_NAME_NOT_DEFD

AP_SNA_DEFD_COS_CANT_BE_DELETE

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_DLC

DELETE_DLC は、DLC をリセットする場合に、その DLC と関連Uけられたすべてのポート、リンク・ステーション、および接続ネットワーク伝送グループ (TG) を削除します。DLC 制御ブロックがすべて削除され、メモリーが解放されます。ノード・オペレーター機能は応答を戻し、DLC が正常に削除されたかどうかを示します。

特定の PU と関連Uけられているリンク・ステーションが（DLC と関連Uけられているために）削除される場合、この PU 上で定義された LU もすべて削除されてしまうことに注意してください。

## VCB 構造体

```
typedef struct delete_dlc
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   attributes;         /* verb attributes              */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   dlc_name[8];        /* name of DLC                  */

} DELETE_DLC;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_DELETE_DLC

**attributes**
verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**dlc_name**
削除する DLC の名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DLC_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_DLC_ACTIVE

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_DOWNSTREAM_LU

この verb はCommunications Serverにのみ適用します。

## VCB 構造体

```
typedef struct delete_downstream_lu
{
   unsigned short  opcode;                  /* verb operation code          */
   unsigned char   attributes;              /* verb attributes              */
   unsigned char   format;                  /* format                       */
   unsigned short  primary_rc;              /* primary return code          */
   unsigned long   secondary_rc;            /* secondary return code        */
        unsigned char   dslu_name[8];       /* Downstream LU name           */
} DELETE_DOWNSTREAM_LU;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_DOWNSTREAM_LU

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

ビット単位で OR 結合されてこのフィールドに入る他の値は次のとおりです。

**AP_DELAY_IF_REQUIRED**

これは、**dslu_name** によってX定されるダウンストリーム LU は現在アクティブで、この verb は LU が非アクティブになるまで 「プログラム」の内 t にキューされるべきであることをX定します。このケースでは、verb は LU が非アクティブになったときに完了に向けてプロセスされます。

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**dslu_name**

削除するダウンストリーム LU の名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_DSLU_ACTIVE
AP_DELAYED_VERB_PENDING

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_DOWNSTREAM_LU_RANGE

この verb はCommunications Serverにのみ適用します。

たとえば、1 から 4 の NAU 範囲で結合された LUNME のベース名は、LUNME001、LUNME002、LUNME003、および LUNME004 の LU を削除します。5 未満の非埋め込み文 z で構成されたベース名を定義すると、LU 名は 8 未満の非埋め込み文 z で構成されます。

この verb は、範囲内にあるすべての LU を削除します。この範囲内に LU が存在しない場合、verb はN実に存在する次の範囲内の LU を削除します。X定した範囲内に LU が存在しない場合にのみ、verb は失敗します。

## VCB 構造体

```
typedef struct delete_downstream_lu_range
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   dslu_base_name[5];   /* Downstream LU base name       */
        unsigned char   min_nau;             /* min NAU address in range      */
        unsigned char   max_nau;             /* max NAU address in range      */
} DELETE_DOWNSTREAM_LU_RANGE;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_DOWNSTREAM_LU_RANGE

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**dslu_base_name**

ダウンストリーム LU 名の範囲のベース名。これは、5 バイトのタイプ A の EBCDIC ストリング（文 z で + O）による英数 z であり、右側には EBCDIC スペースを埋め込みます。このベース名は、3 文 z のタイプ A の EBCDIC 数値文 z で構成されて 10 進数の NAU アドレスを示すものであり、これが NAU 範囲内の F LU に追加されます。

**min_nau**
この範囲における NAU アドレスの最小値。この値は、包 g 的に 1 から 255 とすることができます。

**max_nau**
この範囲における NAU アドレスの最大値。この値は、包 g 的に 1 から 255 とすることができます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_NAU_ADDRESS

AP_INVALID_LU_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

AP_INVALID_LU_NAME
AP_DSLU_ACTIVE
AP_DELAYED_VERB_PENDING

**secondary_rc**
AP_INVALID_LU_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_DSPU_TEMPLATE

この verb はCommunications Serverにのみ適用します。

## VCB 構造体

### Format 1

```
typedef struct delete_dspu_template
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   template_name[8];    /* name of template              */
        unsigned short  num_of_dslu_templates;
                                             /* Number of DSLU templates      */
        unsigned char   reserv1[10];         /* reserved                      */
} DELETE_DSPU_TEMPLATE;

typedef struct dslu_template
{
        unsigned char   min_nau;             /* min NAU address in range      */
        unsigned char   max_nau;             /* max 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;
```

## VCB 構造体

### Format 0

```
typedef struct delete_dspu_template
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   template_name[8];    /* name of template              */
} DELETE_DSPU_TEMPLATE;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_DSPU_TEMPLATE

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**template_name**
DSPU テンプレートの名前（これは、PORT_DEF_DATA の **implicit_dspu_template** フィールドでX定した名前に対応しています)。これは、ローカルに = 示可能な文 z セットによる 8 バイトのストリングです。 8 バイトすべてが有効であり、すべて設定する , 要があります。

**num_of_dslu_templates**
DEFINE_DSPU_TEMPLATE VCB に続く DSLU テンプレート・オーバーレーの数。この値は、包 g 的に 0 から 255 とすることができます。DSLU テンプレートは DELETE_DSPU_TEMPLATE VCB の終わりにオーバーレーとしてU加されます。

**dslu_template.min_nau**
この範囲における NAU アドレスの最小値。この値は、包 g 的に 1 から 255 とすることができます。

**dslu_template.max_nau**
この範囲における NAU アドレスの最大値。この値は、包 g 的に 1 から 255 とすることができます。

**def_data.allow_timeout**
このフィールドは予約済みです。

**def_data.delayed_logon**
このフィールドは予約済みです。

**dslu_template.host_lu**
このフィールドは予約済みです。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TEMPLATE_NAME

AP_INVALID_NAU_RANGE

関係のある START_NODE パラメーターがセットされなかったために verb が実行されない場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_FOCAL_POINT

DELETE_FOCAL_POINT verb をH用して、X定したタイプおよびカテゴリーのフォーカル・ポイントを削除できます。フォーカル・ポイントのタイプの詳細は、68ページの『DEFINE_FOCAL_POINT』を 2 照してください。 h 動状態のフォーカル・ポイントを削除すると、そのフォーカル・ポイントは取り消されます。(任意のタイプの） h 動状態にあるフォーカル・ポイントを取り消すには、AP_ACTIVE のタイプをX定してください。 （AP_BACKUP または AP_IMPLICIT をX定して） h 動状態になっていないときにバックアップ・フォーカル・ポイントまたは暗黙のフォーカル・ポイントを削除する場合、それらのフォーカル・ポイントについての情報があればその情報が除去されます。

さらに DEFINE_FOCAL_POINT verb をH用して、現在 h 動状態にあるフォーカル・ポイントを取り消すこともできることに注意してください。この二重の機能は、互換性が保たれるようになっています。

## VCB 構造体

```
typedef struct delete_focal_point
{
        unsigned short  opcode;             /* verb operation code            */
        unsigned char   reserv2;            /* reserved                       */
        unsigned char   format;             /* format                         */
        unsigned short  primary_rc;         /* primary return code            */
        unsigned long   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;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_FOCAL_POINT

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**ms_category**

管理サービスのカテゴリー。これは、「SNA 管理サービス」で説明されている管理サービス・カテゴリーの 4 バイトの体系定義値（右側に EBCDIC スペースが埋め込まれている）の 1 つとするか、8 バイトのタイプ 1134 EBCDIC 導入定義名のいずれかにできます。

**type** 削除するフォーカル・ポイントのタイプをX定します。X定可能なタイプは、以下のとおりです。

**AP_ACTIVE**

現在 h 動状態にある（任意のタイプの）フォーカル・ポイントを取り消します。

**AP_IMPLICIT**
暗黙の定義を削除します。現在 h 動状態にあるフォーカル・ポイントが暗黙のフォーカル・ポイントである場合、そのフォーカル・ポイントが取り消されます。

**AP_BACKUP**
バックアップ定義を削除します。現在 h 動状態にあるフォーカル・ポイントがバックアップ・フォーカル・ポイントである場合、そのフォーカル・ポイントが取り消されます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TYPE

AP_INVALID_CATEGORY_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_INTERNAL_PU

DELETE_INTERNAL_PU verb は、 DLUR 向けローカル PU の削除を要求します。この verb は、 h 動状態にある SSCP-PU セッションが PU にない場合にのみ成功します。

この PU に関連Uけられた LU はすべて削除されます。

## VCB 構造体

```
typedef struct delete_internal_pu
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   attributes;         /* verb attributes               */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   pu_name[8];         /* internal PU name              */
} DELETE_INTERNAL_PU;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_INTERNAL_PU

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**pu_name**

削除する内ｔ PU の名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PU_NOT_RESET

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_LOCAL_LU

DELETE_LOCAL_LU verb は、ローカル LU 定義を削除するよう要求します。

## VCB 構造体

```
typedef struct delete_local_lu
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   lu_name[8];         /* local LU name                 */
} DELETE_LOCAL_LU;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_DELETE_LOCAL_LU

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**
定義するローカル LU の名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_CANT_DELETE_CP_LU

ノードがまだ+Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_LS

DELETE_LS は、リンク・ステーションが以前に定義されリセットされていることを検査します。これによりリンク・ステーションの制御ブロックが除去され、リンク・ステーションが正常に削除されたことを示す応答がノード・オペレーター機能から戻されます。このリンク・ステーションをH用して PU 上で定義された LU があれば、それらも削除されてしまうことに注意してください。

## VCB 構造体

```
typedef struct delete_ls
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   attributes;         /* verb attributes              */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   ls_name[8];         /* name of link station         */
} DELETE_LS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_DELETE_LS

**attributes**
verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**ls_name**
削除するリンク・ステーションの名前。これは、ローカル=示可能文zセットの 8 バイトの文z列です。8 バイトすべてが有効であり、すべて設定する，要があります。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LINK_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_LS_ACTIVE

AP_INVALID_LINK_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_LU_0_TO_3

この verb をH用して、特定の LU を削除します。

## VCB 構造体

```
typedef struct delete_lu_0_to_3
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   attributes;         /* verb attributes              */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   lu_name[8];         /* LU name                      */
} DELETE_LU_0_TO_3;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_LU_0_TO_3

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

削除する LU の名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_LU_NAME

AP_CANT_DELETE_IMPLICIT_LU

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_LU_0_TO_3_RANGE

この verb をH用して、特定範囲の LU を削除します。このノード・オペレーターが、ベース名と NAU 範囲をX定します。LU 名は、ベース名と NAU アドレスを結合することにより生成されます。

たとえば、1 から 4 の NAU 範囲で結合された LUNME のベース名は、LUNME001、LUNME002、LUNME003、および LUNME004 の LU を削除します。 5 未満の非埋め込み文 z で構成されたベース名を定義すると、 LU 名は 8 未満の非埋め込み文 z で構成されます。

この範囲内のすべての LU が削除されます。この範囲内に LU が存在しない場合、verb はN実に存在する次の範囲内の LU を削除します。 X定した範囲内に LU が存在しない場合に、verb は失敗します。

## VCB 構造体

### Format 1

```
typedef struct delete_lu_0_to_3_range
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   attributes;         /* verb attributes              */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   base_name[6];       /* base name                    */
        unsigned char   min_nau;            /* minimum NAU address          */
        unsigned char   max_nau;            /* maximum NAU address          */
        unsigned char   name_attributes;    /* Attributes of base_name      */
        unsigned char   base_number;        /* Base number for LU names     */
        unsigned char   reserv5[16];        /* reserved                     */
} DELETE_LU_0_TO_3_RANGE;
```

## VCB 構造体

### Format 0

```
typedef struct delete_lu_0_to_3_range
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   attributes;         /* verb attributes              */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   base_name[5];       /* base name                    */
        unsigned char   min_nau;            /* minimum NAU address          */
        unsigned char   max_nau;            /* maximum NAU address          */
        unsigned char   reserv3;            /* reserved                     */
} DELETE_LU_0_TO_3_RANGE;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_LU_0_TO_3_RANGE

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**base_name**

ベースの LU 名。これは、5 バイトのタイプ A の EBCDIC ストリング（文 z で + O）による英数 z であり、右側には EBCDIC スペースを埋め込みます。このベース名は、3 文 z のタイプ A の EBCDIC 数値文 z で構成されて 10 進数の NAU アドレスを示すものであり、これが NAU 範囲内の F LU に追加されます。

**min_nau**

この範囲における NAU アドレスの最小値。この値は、包 g 的に 1 から 255 とすることができます。

**max_nau**

この範囲における NAU アドレスの最大値。この値は、包 g 的に 1 から 255 とすることができます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_NAU_ADDRESS

AP_INVALID_LU_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_STATE_CHECK

**secondary_rc**

AP_INVALID_LU_NAME

AP_CANT_DELETE_IMPLICIT_LU

## DELETE_LU_0_TO_3_RANGE

従属 LU サポートをH用してシステムが構築されていないために verb が実行されない場合、「プログラム」は次のパラメーターが戻されます。

**primary_rc**
AP_INVALID_VERB

ノードがまだ+Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停_しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_LU_POOL

この verb をH用して、LU プールを削除するか、プールから LU を除去します。LU 名をX定しない場合、プール全体が除去されます。LU プール内でX定した LU、または LU プールそのものが存在しなくなるときに、この verb は実行が正常に完了します。この verb は、X定した LU が存在しない場合、もしくはX定したプール内に LU が全くない場合に限り、実行が失敗します。

## VCB 構造体

```
typedef struct delete_lu_pool
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   attributes;         /* verb attributes              */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   pool_name[8];       /* LU pool name                 */
        unsigned short  num_lus;            /* number of LUs to add         */
        unsigned char   lu_names[10][8];    /* LU names                     */
} DELETE_LU_POOL;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_LU_POOL

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**pool_name**

LU プールの名前。 8 バイトすべてが有効であり、すべて設定する , 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 z で + O）による英数 z であり、右側には EBCDIC スペースを埋め込みます。

**num_lus**

**lu_names** リストでX定した LU の数。

**lu_names**

除去する LU の名前。それぞれの名前は、8 バイトのタイプ A の EBCDIC ストリング（文 z で + O）による英数 z であり、右側には EBCDIC スペースを埋め込みます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_POOL_NAME

AP_INVALID_LU_NAME
AP_INVALID_NUM_LUS

従属 LU サポートをH用してシステムが構築されていないために verb が実行されない場合、「プログラム」は次のパラメーターが戻されます。

**primary_rc**
AP_INVALID_VERB

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_MODE

DELETE_MODE verb は、モード定義を削除するよう要求します。
CPSVCMG、SNASVCMG、およびその他の 8 準 SNA モードのデフォルト定義は削除されません。

## VCB 構造体

```
typedef struct delete_mode
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   mode_name[8];       /* mode name                     */
} DELETE_MODE;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_DELETE_MODE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**mode_name**
モードの名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_CP_OR_SNA_SVCMG_UNDELETABLE

AP_MODE_UNDELETABLE
AP_DEL_MODE_DEFAULT_SPCD
AP_MODE_NAME_NOT_DEFD

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_PARTNER_LU

DELETE_PARTNER_LU は、パートナー LU 定義を削除するよう要求します。

## VCB 構造体

```
typedef struct delete_partner_lu
{
        unsigned short  opcode;             /* verb operation code            */
        unsigned char   reserv2;            /* reserved                       */
        unsigned char   format;             /* format                         */
        unsigned short  primary_rc;         /* primary return code            */
        unsigned long   secondary_rc;       /* secondary return code          */
        unsigned char   fqplu_name[17];     /* fully qualified partner        */
                                            /* LU name                        */
} DELETE_PARTNER_LU;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_DELETE_PARTNER_LU

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**fqplu_name**
パートナー LU の完全修飾名。この名前の長さは 17 バイトで、 EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文z の文z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PLU_NAME

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_PORT

DELETE_PORT は、ポートがリセットされる場合にそのポートに関連U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伝送グループ (TG) を削除します。その後、ポートの制御ブロックを削除し、メモリーを解放してから、ポートが正常に削除されたかどうかを示すノード・オペレーター機能からの応答が戻されます。

特定の PU と関連Uけられているリンク・ステーションが（ポートと関連Uけられているために）削除される場合、この PU 上で定義された LU もすべて削除されてしまうことに注意してください。

## VCB 構造体

```
typedef struct delete_port
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   attributes;         /* verb attributes              */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   port_name[8];       /* name of port                 */
} DELETE_PORT;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_PORT

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**port_name**

削除するポートの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PORT_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PORT_ACTIVE

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_TP

DELETE_TP は、トランザクション・プログラム (TP) の定義を削除するよう要求します。

## VCB 構造体

```
typedef struct delete_tp
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   attributes;         /* verb attributes               */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   tp_name[64];        /* TP name                       */
} DELETE_TP;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_TP フォーマットは、VCB の形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**tp_name**

トランザクション・プログラムの名前。「プログラム」はこのフィールドの文 z セットを検査しません。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_TP_NAME

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# 第5章 活動化と非活動化 verb

この章では、以下のものをh動化または非h動化する際にH用する verb について説明します。

- データ・リンク制御 (DLC)
- 内t PU
- ポート
- リンク・ステーション
- セッション
- 会話グループ

また、高性能経路X定 (HPR) をサポートする接続へのパス・スイッチを要求するときにH用する verb についても解説します。

# START_DLC

START_DLC はデータ・リンク制御 (DLC) の h 動化を要求します。その後に、DLC の h 動化が成功したかどうかを示す情報が戻されます。 DLC 用に定義されているポートがなくても、DLC を + O することは可能ですので注意してください。 DLC、ポート、およびリンク・ステーションの関連についての詳細は、15ページの『DLC プロセス、ポート、リンク・ステーション』を 2 照してください。

## VCB 構造体

```
typedef struct start_dlc
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   dlc_name[8];        /* name of DLC                   */
} START_DLC;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターを X 定します。

**opcode**
AP_START_DLC

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**dlc_name**
+ O するデータ・リンク制御インスタンスの名前。これは、ローカルに = 示可能な文 z セットによる 8 バイトのストリングです。 DEFINE_DLC verb によってすでに定義されていなければなりません。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DLC

DLC が非 h 動化中であるために verb の処理が実行されなかった場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_DLC_DEACTIVATING

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# START_INTERNAL_PU

START_INTERNAL_PU verb は従属型 LU リクエスター (DLUR) に対して、DLUR からサービスを提供されており直前に定義したローカル PU の SSCP-PU セッションの h 動化を + O するように要求します。

## VCB 構造体

```
typedef struct start_internal_pu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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;
```

## 指定 Q i a ー ? ー

アプリケーションは以下のパラメーターを X 定します。

**opcode**

AP_START_INTERNAL_PU

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**pu_name**

SSCP-PU セッションの h 動化フローが送信請求される内 t PU の名前。 これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**dlus_name**

DLUR が特定の PU の SSCP-PU セッション h 動化を送信請求することを連絡する従属型 LU サーバー (DLUS) ノードの名前。この名前は、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文 z ストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。）この値は、DEFINE_INTERNAL_PU verb に X 定された値を X 定変更します。このフィールドをゼロに設定した場合、DEFINE_INTERNAL_PU verb に X 定された DLUS が H 用されます。 DEFINE_INTERNAL_PU verb に DLUS が X 定されていない場合は、グローバル省略 ~ 値（DEFINE_DLUR_DEFAULTS verb によって X 定されている場合）が H 用されます。

**bkup_dlus_name**

特定の 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、すべてゼロに設定されるか、2 つのタイプ A の EBCDIC 文 z ストリングで構成されなければなりません。後者の場合、これらは EBCDIC ドットで連結されており、右側には EBCDIC スペースを埋め込みます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さ

です。）この値は、DEFINE_INTERNAL_PU verb にX定された値をX定変更します。このフィールドがすべてゼロに設定されている場合、DEFINE_INTERNAL_PU verb によってX定された DLUS 名が、この PU のバックアップ DLUS として保 } されます。 DEFINE_INTERNAL_PU verb によってバックアップ DLUS がX定されていない場合、グローバル・バックアップ省略~ DLUS（DEFINE_DLUR_DEFAULTS verb によって定義されている場合）がこの PU のバックアップ省略~値として保 } されます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DLUS_NAME

AP_INVALID_BKUP_DLUS_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_NO_DEFAULT_DLUS_DEFINED

AP_PU_NOT_DEFINED
AP_PU_ALREADY_ACTIVATING
AP_PU_ALREADY_ACTIVE

verb が正常に実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_UNSUCCESSFUL

**secondary_rc**
AP_DLUS_REJECTED

AP_DLUS_CAPS_MISMATCH
AP_PU_FAILED_ACTPU

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# START_LS

START_LS はリンクの h 動化を要求します。リンクが正常に h 動化されたかどうかを X定する応答として戻されます。

DLC、ポート、およびリンク・ステーションの関連についての詳細は、15ページの『DLC プロセス、ポート、リンク・ステーション』を 2 照してください。

## VCB 構造体

```
typedef struct start_ls
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   ls_name[8];         /* name of link station          */
        unsigned char   enable;             /* whether the link is enabled   */
        unsigned char   reserv3[3];         /* reserved                      */
} START_LS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターを X定します。

**opcode**

AP_START_LS

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X定するには、このフィールドをゼロに設定します。

**ls_name**

+ Oするリンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。 **ls_name** の値は、 DEFINE_LS verb での値と同じでなければなりません。

**enable**

リンクを + O する場合に、このフィールドを設定します。このフィールドを AP_ACTIVATE に設定すると、リンクが + O します。この値に設定しない場合、リンクは + O されませんが、以下の値を H用することができます。これらの値は互いに OR 結合できます。

**AP_AUTO_ACT**

リンクはローカル・ノードで , 要になったときに h 動化されます。この値は **auto_act_supp** が DEFINE_LS verb で AP_YES に設定されている場合にのみ有効です。

**AP_REMOTE_ACT**

リンクはリモート・ノードで h 動化されます。この設定は、**disable_remote_act** で定義された値を更新しません。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LINK_NAME_SPECIFIED

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PORT_INACTIVE

AP_ACTIVATION_LIMITS_REACHED
AP_PARALLEL_TGS_NOT_SUPPORTED
AP_ALREADY_STARTING
AP_LINK_DEACT_IN_PROGRESS

後になってリンクがh動化する前に STOP_LS または STOP_PORT によって取り消されたために verb の処理が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_CANCELLED

**secondary_rc**
AP_LINK_DEACTIVATED

リンク・ソフトウェアがパートナーを検出できなかったために verb の処理が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_LS_FAILURE

**secondary_rc**
AP_PARTNER_NOT_FOUND

リンクをN立中にリンク・エラーが起きたために verb の処理が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_LS_FAILURE

**secondary_rc**
AP_ERROR

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# START_PORT

START_PORT は、ポートの h 動化を要求します。ポートが正常に h 動化されたかどうかを示す情報が戻されます。ポート用のリンク・ステーションが定義されていなくてもポートを + O できますが、ポートの親 DLC が非 h 動のときはポートを + O することはできません。

DLC、ポート、およびリンク・ステーションの関連についての詳細は、15ページの『DLC プロセス、ポート、リンク・ステーション』を 2 照してください。

## VCB 構造体

```
typedef struct start_port
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   port_name[8];       /* name of port                  */
} START_PORT;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターを X 定します。

**opcode**
AP_START_PORT

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**port_name**
+ O するポートの名前。ローカルに = 示可能な文 z セットによる 8 バイトのストリングであり、DEFINE_PORT verb での名前と一致していなければなりません。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PORT_NAME

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_DLC_INACTIVE

AP_STOP_PORT_PENDING
AP_DUPLICATE_PORT

verb が取り消されたために verb の処理が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_CANCELLED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# STOP_DLC

STOP_DLC は DLC を停 _ することを要求します。DLC が正常に停 _ したかどうかを示す情報が戻されます。 STOP_DLC は、「プログラム」にX示してこの DLC 上のポートのすべてのリンク・ステーションのh動化の再n行を自動的に停 _ するためにもH用されます。

## VCB 構造体

```
typedef struct stop_dlc
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   stop_type;          /* stop type                     */
        unsigned char   dlc_name[8];        /* name of DLC                   */
} STOP_DLC;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_STOP_DLC

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**stop_type**
DLC を停 _ する方法。

**AP_ORDERLY_STOP**
DLC を停 _ する前に、ノードが終了処理を実行する。

**AP_IMMEDIATE_STOP**
ノードは DLC を即~に停 _ する。

**dlc_name**
停 _ する DLC の名前。ローカルに=示可能な文zセットによる 8 バイトのストリングであり、DEFINE_DLC verb での名前と一致していなければなりません。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_DLC

AP_UNRECOGNIZED_DEACT_TYPE

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_STATE_CHECK

**secondary_rc**

AP_STOP_DLC_PENDING

verb が取り消されたために verb の処理が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_CANCELLED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# STOP_INTERNAL_PU

STOP_INTERNAL_PU verb は、 従属型 LU リクエスター (DLUR) が DLUR によりサービスを提供されている直前に定義したローカル PU の SSCP-PU セッションの非 h 動化を + O するように要求します。

## VCB 構造体

```
typedef struct stop_internal_pu
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   pu_name[8];         /* internal PU name              */
        unsigned char   stop_type;          /* type of stop requested        */
} STOP_INTERNAL_PU;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターを X 定します。

**opcode**
AP_STOP_INTERNAL_PU

**format**
VCB の 形式を識別します。 上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**pu_name**
SSCP-PU セッションを非 h 動化する内 t PU の名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**stop_type**
PU のために要求する停 _ タイプを X 定します。順番に停 _ 処理が行われ、まず基礎となる PLU-SLU および SSCP-LU セッションすべてを非 h 動化してから、 SSCP-PU セッションを非 h 動化します。

AP_ORDERLY_STOP
AP_IMMEDIATE_STOP

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_STOP_TYPE

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PU_NOT_DEFINED

AP_PU_ALREADY_DEACTIVATING
AP_PU_NOT_ACTIVE

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# STOP_LS

STOP_LS は、リンク・ステーションの非 h 動化を要求します。リンクが正常に停 _ したかどうかを示す情報が戻されます。 STOP_LS はリンク・ステーションのリモートからの h 動化を、またはリンク・ステーションのオンデマンド h 動化を H 用 T 可にするためにも H 用できます。 STOP_DLC は、「プログラム」に X 示してすべてのリンク・ステーションの h 動化の再 n 行を自動的に停 _ するためにも H 用されます。

## VCB 構造体

```
typedef struct stop_ls
{
        unsigned short  opcode;             /* verb operation code            */
        unsigned char   reserv2;            /* reserved                       */
        unsigned char   format;             /* format                         */
        unsigned short  primary_rc;         /* primary return code            */
        unsigned long   secondary_rc;       /* secondary return code          */
        unsigned char   stop_type;          /* stop type                      */
        unsigned char   ls_name[8];         /* name of link station           */
        unsigned char   disable;            /* whether the link is disabled   */
        unsigned char   reserved[3];        /* reserved                       */
} STOP_LS;
```

## 指定 Q i a ー ? ー

アプリケーションは以下のパラメーターを X 定します。

**opcode**

AP_STOP_LS

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**stop_type**

リンク・ステーションを停 _ する方法。

**AP_ORDERLY_STOP**

リンク・ステーションを停 _ する前に、ノードが終了処理を実行する。

**AP_IMMEDIATE_STOP**

ノードはリンク・ステーションを即 ~ に停 _ する。

**ls_name**

停 _ するリンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する , 要があります。 **ls_name** の値は、DEFINE_LS verb での値と同じでなければなりません。

**disable**

このリンク・ステーションのリモート h 動化または要求 ~ h 動化を H 用 T 可にするかどうかを X 定します。 AP_NO に設定した場合、リンク・ステーションは DEFINE_LS verb からの **auto_act_supp** および **disable_remote_act** の値によって X 定された状態に戻されます。 それ以 0 の場合は、以下の値を X 定することができます。(以下の値を互いに OR 結合することもできます。)

**AP_AUTO_ACT**
リンクはローカル・ノードで , 要になっても再 h 動化できません。

**AP_REMOTE_ACT**
リンクはリモート・ノードから h 動化できません。
**disable_remote_act** が AP_YES に設定された構成のリンクの場合、このビットは無 k されます（リモート・ノードからの h 動化は、常に STOP_LS によって H 用 T 可にされます）。

**disable** フィールドを AP_NO に設定していない場合、 h 動中ではないまたは非 h 動化の処理中であるリンクに対して、**disable** フィールドを設定するために、 STOP_LS を発行することができます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_UNRECOGNIZED_DEACT_TYPE

AP_LINK_NOT_DEFD

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_LINK_DEACT_IN_PROGRESS

verb が取り消されたために verb の処理が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_CANCELLED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# STOP_PORT

STOP_PORT はポートを停 _ することを要求します。ポートが正常に停 _ したかどうかを示す情報が戻されます。 STOP_PORT は、「プログラム」に X 示してすべてのリンク・ステーションの h 動化の再 n 行を自動的に停 _ するためにも H 用されます。

## VCB 構造体

```
typedef struct stop_port
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   stop_type;          /* Stop Type                     */
        unsigned char   port_name[8];       /* name of port                  */
} STOP_PORT;
```

## 指定 Q i a ー ? ー

アプリケーションは以下のパラメーターを X 定します。

**opcode**
AP_STOP_PORT

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**stop_type**
ポートを停 _ する方法。

**AP_ORDERLY_STOP**
ポートを停 _ する前に、ノードが終了処理を実行する。

**AP_IMMEDIATE_STOP**
ノードはポートを即 ~ に停 _ する。

**port_name**
停 _ するポートの名前。ローカルに = 示可能な文 z セットによる 8 バイトのストリングであり、 DEFINE_PORT verb での名前と一致していなければなりません。

## 戻り Q i a ー ? ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PORT_NAME

AP_UNRECOGNIZED_DEACT_TYPE

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_STOP_PORT_PENDING

verb が取り消されたために verb の処理が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_CANCELLED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# ACTIVATE_SESSION

ACTIVATE_SESSION verb は、ローカル LU とX定したパートナー LU との間で特定のモードの特性をH用するセッションのh動化を要求します。

## VCB 構造体

### Format 1

```
typedef struct activate_session
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   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 identifier            */
        unsigned char   cnos_permitted;     /* is implicit CNOS              */
                                            /* permitted?                    */
        unsigned char   reserv4[15];        /* reserved                      */
} ACTIVATE_SESSION;
```

### Format 0 (バック・レベル)

```
typedef struct activate_session
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   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 identifier            */
} ACTIVATE_SESSION;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_ACTIVATE_SESSION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドにゼロまたは 1 をセットします。

## ACTIVATE_SESSION

**lu_name**

セッションの h 動化を要求するローカル LU の LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。 このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、 **lu_alias** フィールドが H 用されます。

**lu_alias**

セッションの h 動化を要求するローカル LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を } つので、8 バイトすべてを設定する , 要があります。**lu_alias** と **lu_name** の両方をすべてゼロに設定すると、 verb は制御点と関連 U けられている LU（省略 ~ の LU）に転送されます。

**plu_alias**

ローカル LU がパートナー LU を識別するための名前。この名前は、構成 ~ に X 定したパートナー LU の名前と一致していなければなりません。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する , 要があります。このフィールドをすべてゼロに設定すると、**fqplu_name** フィールドが , 須パートナー LU を X 定するために H 用されます。

**mode_name**

構成 ~ に定義されたネットワーキング特性セットの名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**fqplu_name**

パートナー LU の完全修飾 LU 名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。（それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。）このフィールドは、**plu_alias** フィールドにすべてゼロを設定した場合にのみ有効です。

**polarity**

セッション用に要求された極性。H 用できる値は、以下のとおりです。

AP_POL_EITHER
AP_POL_FIRST_SPEAKER
AP_POL_BIDDER

AP_POL_EITHER が選択された場合には、ACTIVATE_SESSION はファースト・スピーカー・セッションが選択可能であればこれを h 動化し、選択可能でなければビッダー・セッションを h 動化します。 AP_POL_FIRST_SPEAKER または AP_POL_BIDDER の場合には、ACTIVATE_SESSION は要求された極性のセッションが選択可能である場合にのみ成功します。

**cnos_permitted**

このフィールドは AP_YES または AP_NO にセットされることもあります。新規セッションの h 動化が、 X 定されたモードのセッション限度がリセットされたために T 可能であり、かつこのフィールドが AP_YES にセットされて

いる場合には、「プログラム」は暗黙の CNOS 処理を + O してセッション限度を初期化します。この verb の実行は CNOS 処理が行われる間は中断状態です。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

**secondary_rc**
AP_AS_SPECIFIED

AP_AS_NEGOTIATED

**session_id**
h 動化セッションの 8 バイト識別 R。

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_EXCEEDS_MAX_ALLOWED

AP_INVALID_CNOS_PERMITTED
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_MODE_NAME
AP_INVALID_PLU_NAME

verb がモードのセッション限度を超えた場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**Secondary_rc**
AP_EXCEEDS_MAX_ALLOWED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

その他のエラーのために verb が実行されない場合、「プログラム」は以下のパラメーターのいずれか 1 つを戻します。

**primary_rc**

AP_ACTIVATION_FAIL_NO_RETRY

AP_ACTIVATION_FAIL_RETRY

# DEACTIVATE_CONV_GROUP

DEACTIVATE_CONV_GROUP は、X定した会話グループに関係するセッションの非h動化を要求します。この verb はノード・オペレーター機能 API の一 t ですが、アプリケーション・プログラマーがパーソナル・コミュニケーションズまたは Communications Server APPC API を用いたトランザクション・プログラムの作成にH用すること主に意図したものです。会話グループ識別Rは、 *パーソナル・コミュニケーションズ・クライアント/サーバー・コミュニケーションズ・プログラミング* で定義された MC_ALLOCATE、ALLOCATE、MC_GET_ATTRIBUTES、GET_ATTRIBUTES および RECEIVE_ALLOCATE の F verb によって戻されます。

## VCB 構造体

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

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEACTIVATE_CONV_GROUP

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

会話グループの非h動化を要求するローカル LU の LU 名。この名前は、8バイトのタイプ A の EBCDIC 文z列です。このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、 **lu_alias** フィールドがH用されます。

**lu_alias**

会話グループの非h動化を要求する ローカル LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。 このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を } つので、8 バイトすべてを設定する , 要があります。**lu_name** フィールドと **lu_alias** フィールドを両方ともすべてゼロに設定すると、 verb は、制御点に関連Uけられている LU（省略~の LU）に転送されます。

**conv_group_id**
セッションが非 h 動化される会話グループ識別 R。

**type** 非 h 動化のタイプ。このフィールドは、verb が非同期的に完了するか、または同期的に完了するかを示すフラグと OR 結合された非 h 動化タイプから構成されるビット・マスクです。

非 h 動化タイプは次のとおりです。

**AP_DEACT_CLEANUP**
セッションは、パートナー LU からの応答を待たずに即 ~ に終了します。

**AP_DEACT_NORMAL**
セッションを H 用しているすべての会話が終わってから、セッションを終了します。

verb の振る q いは次のとおりです。

**AP_ASYNCHRONOUS_DEACTIVATION**
verb は即 ~ に戻ります。

**AP_SYNCHRONOUS_DEACTIVATION**
verb はセッションが非 h 動化してから戻ります。

**sense_data**
非 h 動化処理の CLEANUP タイプに H 用するセンス・データを X 定します。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_CLEANUP_TYPE

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEACTIVATE_SESSION

DEACTIVATE_SESSION verb は、特定のセッションの非 h 動化、または特定のモードのセッションすべての非 h 動化を要求します。

## VCB 構造体

```
typedef struct deactivate_session
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   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                      */
        unsigned long   sense_data;         /* deactivation sense data       */
        unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                            /* LU name                       */
        unsigned char   reserv4[20];        /* reserved                      */
} DEACTIVATE_SESSION;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEACTIVATE_SESSION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

セッションの非 h 動化を要求するローカル LU の LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、**lu_alias** フィールドがH用されます。

**lu_alias**

セッションの非 h 動化を要求するローカル LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を } つので、8 バイトすべてを設定する , 要があります。**lu_name** フィールドと **lu_alias** フィールドを両方ともすべてゼロに設定すると、 verb は、制御点に関連Uけられている LU（省略 ~ の LU）に転送されます。

**session_id**

非 h 動化するセッションの 8 バイト識別R。このフィールドをすべてゼロに設定すると、パーソナル・コミュニケーションズまたは Communications Server はパートナー LU とモードのセッションすべてを非 h 動化します。

**plu_alias**

ローカル LU がパートナー LU を識別するための名前。この名前は、構成～にX定したパートナー LU の名前と一致していなければなりません。これは、ローカル＝示可能文ｚセットの 8 バイトの文ｚ列です。 8 バイトすべてが有効であり、すべて設定する，要があります。このフィールドをすべてゼロに設定すると、**fqplu_name** フィールドが，須パートナー LU をX定するためにH用されます。

**mode_name**

構成～に定義されたネットワーキング特性セットの名前。これは、8 バイト、英数ｚ、タイプ A の EBCDIC 文ｚ列（先頭は文ｚ）です。EBCDIC スペースが右の余白に埋め込まれます。

**type** 非 h 動化のタイプ。このフィールドは、verb が非同期的に完了するか、または同期的に完了するかを示すフラグと OR 結合された非 h 動化タイプから構成されるビット・マスクです。

非 h 動化タイプは次のとおりです。

**AP_DEACT_CLEANUP**

セッションは、パートナー LU からの応答を待たずに即～に終了します。

**AP_DEACT_NORMAL**

セッションをH用しているすべての会話が終わってから、セッションを終了します。

verb の振るｑいは次のとおりです。

**AP_ASYNCHRONOUS_DEACTIVATION**

verb は即～に戻ります。

**AP_SYNCHRONOUS_DEACTIVATION**

verb はセッションが非 h 動化してから戻ります。

**sense_data**

非 h 動化処理の CLEANUP タイプにH用するセンス・データをX定します。

**fqplu_name**

パートナー LU の完全修飾 LU 名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。 EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文ｚの文ｚ列で構成されます。（それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。）このフィールドは、**plu_alias** フィールドにすべてゼロを設定した場合にのみ有効です。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

**session_id** がどの既存セッションとも一致しない場合、その理由はX定したセッションがすでに非 h 動化されているためであるとWわれます。このとき、verb は正常に終了します。

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_MODE_NAME

AP_INVALID_PLU_NAME
AP_INVALID_CLEANUP_TYPE
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# PATH_SWITCH

PATH_SWITCH verb は、パーソナル・コミュニケーションズまたは Communications Server に対して、高性能経路X定 (HPR) をサポートする接続で経路を切り替えることを要求します。 よりよいパスが検出できない場合、接続は変更されません。

## VCB 構造体

```
typedef struct path_switch
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   rtp_connection_name[8];
                                             /* RTP connection name           */
} PATH_SWITCH;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_PATH_SWITCH

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**rtp_connection_name**
パス・スイッチへの RTP 接続を識別します。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_RTP_CONNECTION

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PATH_SWITCH_IN_PROGRESS

パス・スイッチの n 行が失敗したために verb が実行されない場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNSUCCESSFUL

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# 第6章 照会 verb

この章では、ノードの構成や状態についての情報を照会するときにH用する verb について説明します。

SNA API クライアントでは、特定のパラメーターのみがサポートされます。

詳細については、この章の メモ帳 を2照してください。

# QUERY_ADJACENT_NN

この verb は Communications Server にのみ適用されます。

QUERY_ADJACENT_NN は、ネットワーク・ノードでのみH用され、隣接ネットワーク・ノード (つまり、 CP-CP セッションがアクティブになっている、またはこれまでアクティブになっていた、あるいはしばらくの間アクティブになっていたネットワーク・ノード) に関する情報を戻します。

隣接ノードの情報は、形式化されたリストとして戻されます。特定のネットワーク・ノードに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **adj_nncp_name** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**adj_nncp_name** で配列されています。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。 AP_LIST_FROM_NEXT を選択すると、リストは、定義された配列に従って、 X定の項目 (存在していてもいなくても) の次の項目から + Oされます。

## VCB 構造体

```
typedef struct query_adjacent_nn
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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 adj network node   */
} QUERY_ADJACENT_NN;

typedef struct adj_nncp_data
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   adj_nncp_name[17];   /* CP name of adj. network node  */
        unsigned char   cp_cp_sess_status;   /* CP-CP session status          */
        unsigned long   out_of_seq_tdus;     /* out of sequence TDUs          */
        unsigned long   last_frsn_sent;      /* last FRSN sent                */
        unsigned long   last_frsn_rcvd;      /* last FRSN received            */
        unsigned char   reserva[20];         /* reserved                      */
} ADJ_NNCP_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_ADJACENT_NN

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

リスト情報に何を戻すかを示します。つまり、X定された **adj_nncp_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から + O されます。

**adj_nncp_name**

隣接ネットワーク・ノードの 17 バイトの完全修飾名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

**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，要になるバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**adj_nncp_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**adj_nncp_data.adj_nncp_name**
隣接ネットワーク・ノードの 17 バイトの完全修飾 CP 名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**adj_nncp_data.cp_cp_sess_status**
CP-CP セッションの状況。これは、以下のいずれかに設定されます。

AP_ACTIVE
AP_CONWINNER_ACTIVE
AP_CONLOSER_ACTIVE
AP_INACTIVE

**adj_nncp_data.out_of_seq_tdus**
このノードから受信した out_of_sequence TDU の数。

**adj_nncp_data.last_frsn_sent**
このノードに送信した最終フロー縮小順序番号。

**adj_nncp_data.last_frsn_rcvd**
このノードから受信した最終フロー縮小順序番号。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_ADJ_NNCP_NAME

AP_INVALID_LIST_OPTION

ノードが + O されていないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_ADJACENT_NODE

QUERY_ADJACENT_NODE は、DEFINE_ADJACENT_NODEに構成された隣接ノードに関する情報を戻します。

情報は番号Uきリストに戻されます。リスト内のF項目は、隣接 CP に関する情報が含まれている ADJACENT_NODE_DATA オーバーレーと、その後に続く、隣接 CP と関連したF LU ごとの ADJACENT_NODE_LU_DATA オーバーレーからなっています。

項目は **cp_name** 順に配列され、次に **fqlu_name** 順に配列されます。配列はまず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、 ASCII の辞書配列の順序で行われます (8準の MIB 配列に準拠)。

AP_LIST_FROM_NEXT を選択すると、リストは定義済みの順序に従って次の項目から+Oされます (X定された項目が存在するかしないかに関係なく)。

## VCB 構造体

```
typedef struct query_adjacent_node
{
        unsigned short  opcode;              /* Verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* Primary return code            */
        unsigned long   secondary_rc;        /* Secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  total_num_entries;   /* total number of entries        */
        unsigned char   list_options;        /* listing options                */
        unsigned char   reserv3;             /* reserved                       */
        unsigned char   cp_name[17];         /* CP name of adjacent node       */
} QUERY_ADJACENT_NODE;

typedef struct adjacent_node_data
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned short  sub_overlay_size;    /* size of this stub entry        */
        unsigned char   cp_name[17];         /* CP name                        */
        DESCRIPTION     description;         /* resource description           */
        unsigned char   reserv3[19];         /* reserved                       */
        unsigned short  num_of_lus;          /* number of LUs                  */
} ADJACENT_NODE_DATA;

typedef struct cn_det_data
{
        unsigned short  num_act_ports;       /* number of active ports         */
        unsigned char   reserva[20];         /* reserved                       */
} CN_DET_DATA;

typedef struct cn_def_data
{
        unsigned char   description[RD_LEN]; /* resource description           */
        unsigned char   num_ports;           /* number of ports on CN          */
        unsigned char   reserv1[16];         /* reserved                       */
        TG_DEFINED_CHARS tg_chars;           /* TG characteristics             */
} CN_DEF_DATA;
```

```
typedef struct adjacent_node_lu_data
{
        unsigned short  overlay_size;        /* effective capacity            */
        unsigned char   reserve2[2];         /* reserved                      */
        ADJACENT_NODE_LU adj_lu_def_data;    /* Adjacent LU defined data      */
} ADJACENT_NODE_LU_DATA;

typedef struct adjacent_node_lu
{
        unsigned char   wildcard_lu;         /* Is this LU a wildcard?        */
        unsigned char   fqlu_name[17];       /* Fully-Qualified LU name       */
        unsigned char   reserve1[6];         /* reserved                      */
        ADJACENT_NODE_LU  adj_lu_def_data;   /* Adjacent LU defined data      */
} ADJACENT_NODE_LU;
```

## 指定Ｑｉａ一？一

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_ADJACENT_NODE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

これは、リスト情報に何を戻すかを示します。 **cp_name** のX定 (以下のパラメーターを2照) は、戻される実際の情報の+O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無kされ、戻りリストは、「プログラム」によって維}されるディレクトリーの中の最初の隣接ノードから+Oされます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から+Oされます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から+Oされます。

**cp_name**

隣接ノードの完全修飾名。この名前は、 1 つの EBCDIC ドットで連結され

た 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要になるバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**adjacent_node_data.overlay_size**
この項目内のバイトの数。任意の ADJACENT_NODE_LU_DATA 構造体が含まれており、戻された次の項目に対するオフセット (その項目が存在する場合) です。

**adjacent_node_data.sub_overlay_size**
この項目のノード t 分のバイトの数。 ADJACENT_NODE_LU_DATA 構造体は含まれておらず、この項目の最初の ADJACENT_NODE_LU_DATA フィールドに対するオフセットです。

**adjacent_node_data.cp_name**
\ 隣接オフセ

**adjacent_node_lu_data.adj_lu_def_data.wildcard_lu**
LU 名がワイルドカードとして定義されているかどうかを示します。

**adjacent_node_lu_data.adj_lu_def_data.fqlu_name**
隣接ノードの完全修飾名。この名前の長さは 17 バイトであり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_CP_NAME

AP_INVALID_LIST_OPTION

ノードが + O されていないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_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、『決定済みデータ』 (実行中に動的に収集されたデータ) および 『定義済みデータ』 (DEFINE_CN のアプリケーションによって提供されたデータ) として構造化されます。

この情報は定様式リストとして戻されます。特定の CN に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **fqcn_name** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。このリスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**fqcn_name** で配列されています。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます ( 8 準の MIB 配列に準拠)。

AP_LIST_FROM_NEXT を選択すると、リストは定義済みの順序に従って次の項目から + O されます (X定された項目が存在するかしないかに関係なく)。

## VCB 構造体

```
typedef struct query_cn
{
        unsigned short  opcode;              /* Verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* Primary return code           */
        unsigned long   secondary_rc;        /* Secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this 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
{
        unsigned short  num_act_ports;       /* number of active ports        */
        unsigned char   reserva[20];         /* reserved                      */
} CN_DET_DATA;

typedef struct cn_def_data
{
        unsigned char   description[RD_LEN]; /* resource description          */
```

```
        unsigned char   num_ports;          /* number of ports on CN         */
        unsigned char   reserv1[16];        /* 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;        /* modem class                   */
        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;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_CN

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。つまり、 **fqcn_name** のX定 (以下のパラメーターを 2 照) は、戻される実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**fqcn_name**
完全修飾の 17 バイトの接続ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**cn_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**cn_data.fqcn_name**
完全修飾の 17 バイトの接続ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**cn_data.det_data.num_act_ports**
接続ネットワーク上のアクティブ・ポートの数を示す動的数値。

**cn_data.def_data.description**
q 源の説明 (DEFINE_CN で定義)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**cn_data.def_data.num_ports**
接続ネットワーク上のポートの数。

**cn_data.def_data.tg_chars.effect_cap**
有効な容量を示す実際の単位。この値は、1 バイトの b 動小数点数値としてエンコードされ、公式 0.1mmm * 2 eeeee で = されます。ここで、バイトのビット = 示は eeeeemmm です。有効な容量を示す F 単位は、1 C 当たり 300 ビットになります。

**cn_data.def_data.tg_chars.connect_cost Cost per connect time.**
有効値は、0 ～ 255 の範囲の整数値です。ここで、 0 は接続 ~ 間当たりの最小コスト、255 は最大コストです。

**cn_data.def_data.tg_chars.byte_cost**
バイト当たりのコスト。有効値は、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、255 は最大コストです。

**cn_data.def_data.tg_chars.security**
以下のリストに示されている機密保護値。

**AP_SEC_NONSECURE**
機密保護はありません。

**AP_SEC_PUBLIC_SWITCHED_NETWORK**
この接続ネットワークを介して伝送されるデータは、公衆交換ネットワークを介して流れます。

**AP_SEC_UNDERGROUND_CABLE**
データは、保護地下ケーブルで伝送されます。

**AP_SEC_SECURE_CONDUIT**
この回線は、保護されていない保護コンジットです。

**AP_SEC_GUARDED_CONDUIT**
コンジットは、物理盗聴から保護されています。

**AP_SEC_ENCRYPTED**
回線を介した暗号化。

**AP_SEC_GUARDED_RADIATION**
この回線は、物理・放射盗聴から保護されています。

**cn_data.def_data.tg_chars.prop_delay**
信号がこのリンクの長さを進むのに要する ~ 間を = す伝搬遅延 (マイクロ C 単位)。この値は、1 バイトの b 動小数点数値としてエンコードされ、公式 0.1mmm * 2 eeeee で = されます。ここで、バイトのビット = 示は eeeeemmm です。省略 ~ 値は次のとおりです。

**AP_PROP_DELAY_MINIMUM**
伝搬遅延なし。

**AP_PROP_DELAY_LAN**
480 マイクロ C 未満の遅延。

**AP_PROP_DELAY_TELEPHONE**
480 から 49 512 マイクロ C の間の遅延。

**AP_PROP_DELAY_PKT_SWITCHED_NET**
49 512 から 245 760 マイクロ C の間の遅延。

**AP_PROP_DELAY_SATELLITE**
245 760 マイクロ C より長い遅延。

**AP_PROP_DELAY_MAXIMUM**
伝搬遅延の最大値。

**cn_data.def_data.tg_chars.modem_class**
予約済み。このフィールドは、常にゼロに設定しておかなければなりません。

**cn_data.def_data.tg_chars.user_def_parm_1**
0 ～ 255 の範囲のユーザー定義パラメーター。

**cn_data.def_data.tg_chars.user_def_parm_2**
0 ～ 255 の範囲のユーザー定義パラメーター。

**cn_data.def_data.tg_chars.user_def_parm_3**
0 ～ 255 の範囲のユーザー定義パラメーター。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_CN_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_CN_PORT

QUERY_CN_PORT は、隣接接続ネットワーク上で定義されているポートの情報を戻します。この情報は定様式リストとして戻されます。特定のアプリケーションに関する情報またはいくつかの 『chunks』 内のリスト情報を得るには、 **port_name** フィールドを設定してください。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。 **fqcn_name** フィールドには、常に、有効接続ネットワークの名前を設定しておかなければならない点に注意してください。

リスト形式のH用方法に関する2考情報については、 10ページの『ノードの照会』 を2照してください。

## VCB 構造体

```
typedef struct query_cn_port
{
        unsigned short  opcode;             /* Verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* Primary return code           */
        unsigned long   secondary_rc;       /* Secondary return code         */
        unsigned char   *buf_ptr;           /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;       /* size of this 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;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_CN_PORT

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
この値は、リスト情報に何を戻すかを示します。つまり、 **fqcn_name** と **port_name** を組み合わせたX定 (以下のパラメーターを2照) は、戻される実際の情報の+O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無kされ、戻りリストはリスト内の最初の項目から+Oされます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から+Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から+Oされます。

**fqcn_name**
完全修飾の 17 バイトの接続ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文z列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) このフィールドは,ず設定しなければなりません。

**port_name**
ローカル=示可能文zセットの 8 バイトの文z列。8 バイトすべてが有効であり、すべて設定する,要があります。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無kされます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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,要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**cn_port_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**cn_port_data.fqcn_name**
完全修飾の 17 バイトの接続ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**cn_port_data.port_name**
ローカル = 示可能文 z セットの 8 バイトのポート名。8 バイトすべてが有効です。

**cn_port_data.tg_num**
X定したポートの伝送グループ番号。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_CN_NAME

AP_INVALID_PORT_NAME
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_CONVERSATION

QUERY_CN_PORT は、X定 LU で実行される会話に関するリスト情報を戻します。特定の会話に関する情報、またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **conv_id** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。 **lu_alias** フィールドを常に設定する , 要があることに注意してください。lu_name は、非ゼロの場合、lu_alias に優先してH用されます。

リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、 **conv_id** 順に配列されます。AP_LIST_FROM_NEXT を選択すると、戻りリストは、索引に従って次の項目から + O されます (X定された項目が存在するかしないかには関係なく)。

## VCB 構造体

```
typedef struct query_conversation
{
        unsigned short  opcode;              /* Verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* Primary return code           */
        unsigned long   secondary_rc;        /* Secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  total_num_entries;   /* total number of entries       */
        unsigned char   list_options;        /* listing options               */
        unsigned char   reserv3;             /* reserved                      */
        unsigned char   lu_name[8];          /* local LU name                 */
        unsigned char   lu_alias[8];         /* local LU alias                */
        unsigned long   conv_id;             /* conversation identifier       */
        unsigned char   session_id[8];       /* session identifier            */
        unsigned char   reserv4[12];         /* reserved                      */
} QUERY_CONVERSATION;

typedef struct conv_summary
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned long   conv_id;             /* conversation identifier       */
        unsigned char   local_tp_name[64];   /* Name of local TP              */
        unsigned char   partner_tp_name[64]; /* Name of partner TP            */
        unsigned char   tp_id[8];            /* TP identifier                 */
        unsigned char   sess_id[8];          /* session identifier            */
        unsigned long   conv_start_time;     /* time conversation was         */
                                             /* started                       */
        unsigned long   bytes_sent;          /* bytes sent so far             */
        unsigned long   bytes_received;      /* bytes received so far         */
        unsigned char   conv_state;          /* conversation state            */
        unsigned char   duplex_type;         /* conversation duplex type      */
} CONV_SUMMARY;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_CONVERSATION

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

これは、リスト情報に何を戻すかを示します。X定された **index** (下記 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から + O されます。

**lu_name**

ローカル LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (先頭は非数 z ) で、右側には EBCDIC スペースが埋め込まれています。

**lu_alias**

ローカル LU がローカル TP によって認識される別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。

**conv_id**

会話 ID。

**session_id**

この値がすべて 2 進ゼロであれば、このフィールドは、戻された会話をフィルター処理するためにH用されます。ゼロでなければ、セッション ID が提供値と一致する会話のみが戻されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**conv_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**conv_summary.conv_id**
会話 ID。

このパラメーターの値は、トランザクション・アクションの起動 ~ に ALLOCATE verb によって戻されたか、あるいは起動されたトランザクション・プログラムの RECEIVE_ALLOCATE によって戻されました。

**conv_summary.local_tp_name**
ローカル・トランザクション・プログラムの名前。

**conv_summary.partner_tp_name**
パートナー・トランザクション・プログラムの名前。この値は、ローカル起動会話にのみ有効です。リモート起動会話の場合は、この値はブランクです。

**conv_summary.tp_id**
トランザクション・プログラムに d り当てられたトランザクション・プログラム ID。この ID は、API スタブまたは NOF トランザクション・プログラム・マネージャーによって d り当てられます。

**conv_summary.sess_id**
この会話に d り振られたセッションの ID。

**conv_summary.conv_start_time**
ノードの + O ~ 間から会話の + O ~ 間までの経過 ~ 間 (センチ C 単位)。

**conv_summary.bytes_sent**
この会話で、これまでに送信されたバイトの数。

**conv_summary.bytes_received**
この会話で、これまでに受信されたバイトの数。

**conv_summary_conv_state**

**conv_id** によって識別された会話の現在の状態。半二重会話の場合は、この値は以下のいずれかになります。

**AP_RESET_STATE**

AP_SEND_STATE
AP_RECEIVE_STATE
AP_CONFIRM_STATE
AP_CONFIRM_SEND_STATE
AP_CONFIRM_DEALL_STATE
AP_PEND_POST_STAT
AP_PEND_DEALL_STATE
AP_END_CONV_STATE
AP_SEND_PENDING_STATE
AP_POST_ON_RECEIPT_STATE

全二重会話の場合は、この値は以下のいずれかになります。

AP_RESET_STATE
AP_SEND_RECEIVE_STATE
AP_SEND_ONLY_STATE
AP_RECEIVE_ONLY_STATE

**conv_summary.duplex_type**

この会話が半二重であるか全二重であるかをX定します。

AP_HALF_DUPLEX
AP_FULL_DUPLEX

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_BAD_CONV_ID

AP_INVALID_LU_ALIAS
AP_INVALID_LU_NAME

ノードが+Oされないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_COS

QUERY_COS は、特定のサービス・クラスに関する経路計；情報を戻します。この情報は定様式リストとして戻されます。特定の COS に関する情報またはいくつかの『固まり』 のリスト情報を入手するには、 **cos_name** フィールドを設定する，要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。このリストは、**cos_name** 順に配列されます。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された順序に従って次の項目から + O されます (X 定された項目が存在するかしないかに関係なく)。

## VCB 構造体

```
typedef struct query_cos
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   cos_name[8];         /* COS name                       */
        unsigned char   description[RD_LEN]; /* resource description           */
        unsigned char   transmission_priority;
                                             /* transmission priority          */
        unsigned char   reserv1;             /* reserved                       */
        unsigned short  num_of_node_rows;    /* number of node rows            */
        unsigned short  num_of_tg_rows;      /* number of TG rows              */
        unsigned long   trees;               /* number of tree caches for COS  */
        unsigned long   calcs;               /* number of route calculations   */
                                             /* for this COS                   */
        unsigned long   rejs;                /* number of route rejects        */
                                             /* for COS                        */
        unsigned char   reserva[20];         /* reserved                       */
} COS_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_COS

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
この値は、リスト情報に何を戻すかを示しています。つまり、X定された **cos_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**cos_name**
サービス・クラス名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**cos_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**cos_data.cos_name**
サービス・クラス名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。

**cos_data.description**
q 源の説明 (DEFINE_COS で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**cos_data.transmission_priority**
伝送優先順位。以下のいずれかの値に設定されます。

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

**cos_data.num_of_node_rows**
この COS のノード行の数。

**cos_data.num_of_tg_rows**
この COS の TG 行の数。

**cos_data.trees**
最後の初期設定以降、この COS について作成された経路ツリー・キャッシュの数。

**cos_data.calcs**
このサービス・クラスを X 定するセッション h 動化要求 (つまり、経路計 ; ) の数。

**cos_data.rejs**
ネットワーク上のこのノードから名前 U き宛先までの受け入れ可能な経路 (X 定されたサービス・クラスを H 用) がないために失敗したセッション h 動化要求の数。経路が受け入れ可能になるのは、それが、X 定されたサービス・クラス (COS)を提供できるアクティブ状態の TG とノードだけで構成されている場合のみです。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_COS_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DEFAULT_PU

QUERY_DEFAULT_PU をH用すれば、ユーザーは、 DEFINE_DEFAULT_PU verb をH用して定義されたデフォルト PU を照会することができます。

## VCB 構造体

```
typedef struct query_default_pu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   def_pu_name[8];      /* default PU name               */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   def_pu_sess[8];      /* PU name of active             */
                                             /* default session               */
        unsigned char   reserv3[16];         /* reserved                      */
} QUERY_DEFAULT_PU;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DEFAULT_PU

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

**def_pu_name**

最新の DEFINE_DEFAULT_PU verb にX定された PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。DEFINE_DEFAULT_PU verb が発行されないと、このフィールドはすべてゼロに設定されます。

**description**

q 源の説明 (DEFINE_DEFAULT_PU でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**def_pu_sess**

現在アクティブになっているデフォルト PU セッションと関連する PU の名前。デフォルト PU が定義されていても、それと関連するセッションがアクティブでなければ、この値は、**def_pu_name** フィールドとは異なったものになります。この場合、パーソナル・コミュニケーションズまたは Communications Server は、定義されたデフォルト PU とのセッションがアク

ティブになるまで、前のデフォルト PU と関連するセッションをH用します。アクティブ PU セッションがなければ、このフィールドはすべてゼロに設定されます。

ノードが+ Oされないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DEFAULTS

QUERY_DEFAULTS をH用すれば、ユーザーは、 DEFINE_DEFAULTS verb をH用して定義されたデフォルトを照会することができます。

## VCB 構造体

```
typedef struct query_defaults
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        DEFAULT_CHARS   default_chars;       /* default information           */
} QUERY_DEFAULTS;

typedef struct default_chars
{
        unsigned char   description[RD_LEN]; /* resource description          */
        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                   */
        unsigned char   limited_timeout;     /* timeout for limited           */
                                             /* sessions                      */
        unsigned char   reserv[244];         /* reserved                      */
} DEFAULT_CHARS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DEFAULTS

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

**default_chars.description**

q源の説明 (DEFINE_DEFAULTS でX定)。これは、ローカル=示可能文zセットの 16 バイトの文z列です。16 バイトすべてが有効です。

**default_chars.mode_name**

最新の DEFINE_DEFAULTS verb でX定されたモードの名前。これは、8 バイト英数zのタイプ A の EBCDIC 文z列 (文zでOまる) で、右側に

EBCDIC スペースが埋め込まれています。DEFINE_DEFAULTS verb が発行されないと、このフィールドはすべてゼロに設定されます。

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DIRECTORY_ENTRY

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。この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の LU に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **resource_name** および **resource_type** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

ローカル・ノードがネットワーク・ノードであれば、情報は次のように戻されます。

最初のネットワーク・ノード
- ネットワーク・ノードに配置された最初の LU
- ネットワーク・ノードに配置された 2 番目の LU
- ...
- ネットワーク・ノードに配置された n 番目の LU

ネットワーク・ノードによって提供される最初のエンド・ノード
- エンド・ノード (1) に配置された最初の LU
- エンド・ノード (1) に配置された 2 番目の LU
- ...
- エンド・ノード (1) に配置された n 番目の LU

...

ネットワーク・ノードによって提供される n 番目のエンド・ノード
- エンド・ノード (n) に配置された最初の LU
- エンド・ノード (n) に配置された 2 番目の LU
- ...

2 番目のネットワーク・ノード

...

「プログラム」をエンド・ノードとして操作すると、 q 源リストに戻される最初の項目は EN CP です。(エンド・ノードのネットワーク・ノード・サーバーの場合は、項目は戻されません。)

戻されたこのディレクトリー項目のリストは、親の名前 (およびタイプ) 別にフィルター処理することができます。この場合は、**parent_name** と **parent_type** の両方のフィールドを設定する , 要があります (そうしない場合は、これらのフィールドをすべてゼロに設定する , 要があります)。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。 AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

## VCB 構造体

**Format 1**

```
typedef struct query_directory_entry{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  total_num_entries;   /* total number of entries       */
        unsigned char   list_options;        /* listing options               */
        unsigned char   reserv3;             /* reserved                      */
        unsigned char   resource_name[17];   /* network qualified res name    */
        unsigned char   reserv4;             /* reserved                      */
        unsigned short  resource_type;       /* Resource type                 */
        unsigned char   parent_name[17];     /* parent name filter            */
        unsigned char   reserv5;             /* reserved                      */
        unsigned short  parent_type;         /* parent type                   */
        unsigned char   reserv6[24];         /* reserved                      */
} QUERY_DIRECTORY_ENTRY;

typedef struct directory_entry_summary
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   resource_name[17];   /* network qualified res name     */
        unsigned char   reserve1;            /* reserved                       */
        unsigned short  resource_type;       /* Resource type                  */
        unsigned char   description[RD_LEN]; /* resource description           */
        unsigned char   real_owning_cp_type; /* real owning CP type            */
        unsigned char   real_owning_cp_name[17];
                                             /* real owning CP name            */
} DIRECTORY_ENTRY_SUMMARY;

typedef struct directory_entry_detail
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   resource_name[17];   /* network qualified res name    */
        unsigned char   reserv1a;            /* reserved                      */
        unsigned short  resource type;       /* Resource type                 */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   parent_name[17];     /* network qualified             */
                                             /* parent name                   */
        unsigned char   reserv1b;            /* reserved                      */
        unsigned short  parent_type;         /* parent resource type          */
        unsigned char   entry_type;          /* Type of the directory entry   */
        unsigned char   location;            /* Resource location             */
        unsigned char   real_owning_cp_type; /* real owning CP type           */
        unsigned char   real_owning_cp_name[17];                              */
                                             /* real owning CP name           */
        unsigned char   reserva;             /* reserved                      */
} DIRECTORY_LU_DETAIL;
```

## VCB 構造体

### Format 0 (バック・レベル)

```
typedef struct query_directory_entry{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
```

```
        unsigned short  total_num_entries;   /* total number of entries       */
        unsigned char   list_options;        /* listing options               */
        unsigned char   reserv3;             /* reserved                      */
        unsigned char   resource_name[17];   /* network qualified res name    */
        unsigned char   reserv4;             /* reserved                      */
        unsigned short  resource_type;       /* Resource type                 */
        unsigned char   parent_name[17];     /* parent name filter            */
        unsigned char   reserv5;             /* reserved                      */
        unsigned short  parent_type;         /* parent type                   */
} QUERY_DIRECTORY_ENTRY;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**
AP_QUERY_DIRECTORY_ENTRY

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。VCB の形式に影響を与えるほか、AP_DLUR_LU_RESOURCE の q 源を戻すのも、Format 1 だけです。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

**resource_name** と **resource_type** を組み合わせたX定 (下記のパラメーターを 2 照) は、戻される実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**resource_name**
ネットワーク修飾 q 源名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**resource_type**
q 源タイプ。以下のいずれかの値になります。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE
AP_DLUR_LU_RESOURCE

**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**parent_name**
親名フィルター。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) このフィールドを設定すると、X定された親に属するディレクトリー項目のみが戻されます (この場合、 **parent_name** フィールドもX定する , 要があります)。すべてゼロに設定すると、このフィールドは無 k されます。

**parent_type**
**parent_name** フィールドにX定された親のタイプ。 **parent_name** フィールドが非ゼロの場合は、タイプをX定する , 要があります。ゼロの場合は、このフィールドをゼロに設定する , 要があります。これは、以下のいずれかに設定することができます。

AP_ENCP_RESOURCE
AP_NNCP_RESOURCE

**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
戻されたディレクトリー項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**directory_entry_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**directory_entry_summary.resource_name**
ネットワーク修飾 q 源名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**directory_entry_summary.resource_type**
q 源タイプ。これは、以下のいずれかにすることができます。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE
AP_DLUR_LU_RESOURCE

(**format** をゼロに設定すると、戻されません。)

**directory_entry_summary.description**
以下のパラメーターにX定された q 源の説明。

DEFINE_LOCAL_LU
DEFINE_DIRECTORY_ENTRY
DEFINE_ADJACENT_LEN_NODE or
DEFINE_ADJACENT_NODE

**directory_entry_summary.real_owning_cp_type**
NN および BrNN のみ: 実所有 CP タイプ。これは、以下のいずれかにすることができます。

**AP_NONE**
実所有 CP は親 q 源です。

**AP_ENCP_RESOURCE**
実所有 CP は親 q 源ではなく、EN です。

その他のノード・タイプ: このフィールドは AP_NONE に設定されます。

**directory_entry_summary.real_owning_cp_name**
NN および BrNN のみ: 完全修飾実所有 CP 名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つの

ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

実所有 CP が親の場合は、このフィールドは 2 進ゼロに設定されます。

実所有 CP が親でない場合は、このフィールドは実所有 CP の名前に設定されます。

q 源が BrNN のドメイン内の EN によって所有されている場合は、実所有 CP は、BrNN の NNS のディレクトリー内の親ではありません。この場合、実所有 CP は EN ですが、親は BrNN です。

その他のノード・タイプ: このフィールドは 2 進ゼロに設定されます。

**directory_entry_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**directory_entry_detail.resource_name**
ネットワーク修飾 q 源名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**directory_entry_detail.resource_type**
q 源タイプ。これは、以下のいずれかにすることができます。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

**directory_entry_detail.description**
以下のパラメーターに X 定された q 源の説明。

DEFINE_LOCAL_LU
DEFINE_DIRECTORY_ENTRY
DEFINE_ADJACENT_LEN_NODE or
DEFINE_ADJACENT_NODE

**directory_entry_detail.parent_name**
LU にサービスを提供するノードの完全修飾親名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**directory_entry_detail.parent_type**
親 q 源タイプ。これは、以下のいずれかにすることができます。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE

**directory_lu_detail.entry_type**
ディレクトリー項目のタイプを X 定します。これは、以下のいずれかの値にすることができます。

**AP_HOME**
ローカル q 源。

**AP_CACHE**
キャッシュ項目。

**AP_REGISTER**
登録済みの q 源 (NN のみ)。

**directory_entry_detail.location**
q 源の位置を示します。値は、以下のいずれかにすることができます。

**AP_LOCAL**
q 源はローカル・ノードにあります。

**AP_DOMAIN**
q 源はU加エンド・ノードに属しています。

**AP_CROSS_DOMAIN**
q 源はローカル・ノードのドメイン内にはありません。

**directory_entry_detail.real_owning_cp_type**
NN および BrNN のみ: 実所有 CP タイプ。これは、以下のいずれかにすることができます。

**AP_NONE**
実所有 CP は親 q 源です。

**AP_ENCP_RESOURCE**
実所有 CP は親 q 源ではなく、EN です。

その他のノード・タイプ: このフィールドは AP_NONE に設定されます。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_RES_NAME

AP_INVALID_RES_TYPE
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DIRECTORY_LU

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。この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の LU に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **lu_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10 ページの『ノードの照会』 を 2 照してください。

このリストは、**lu_name** 順に配列されます。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。 AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

ディレクトリーに含まれている DLUS 提供の LU も、この照会によって戻されることに注意してください。

## VCB 構造体

```
typedef struct query_directory_lu
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   lu_name[17];         /* network qualified LU name      */
        unsigned char   description[RD_LEN]; /* resource description           */
} DIRECTORY_LU_SUMMARY;

typedef struct directory_lu_detail
{
        unsigned short  overlay_size;        /* size of this entry           */
        unsigned char   lu_name[17];         /* network qualified LU name    */
        unsigned char   description[RD_LEN]; /* resource description         */
        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];
                                             /* apparent LU owner name       */
        unsigned char   reserva[3];          /* reserved                     */
} DIRECTORY_LU_DETAIL;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**
AP_QUERY_DIRECTORY_LU

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

X定された **lu_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**lu_name**
ネットワーク修飾 LU 名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
戻されたディレクトリー項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**directory_lu_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**directory_lu_summary.lu_name**
ネットワーク修飾 LU 名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**directory_lu_summary.description**
q 源の説明 (DEFINE_LOCAL_LU または DEFINE_ADJACENT_NODE で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**directory_lu_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**directory_lu_detail.lu_name**
ネットワーク修飾 LU 名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**directory_lu_detail.description**
q 源の説明 (DEFINE_LOCAL_LU または DEFINE_ADJACENT_NODE で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**directory_lu_detail.server_name**
LU にサービスを提供するノードのネットワーク修飾名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1

つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**directory_lu_detail.lu_owner_name**
LU を所有するノードのネットワーク修飾名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**directory_lu_detail.location**
q 源の位置を X 定します。値は、以下のいずれかにすることができます。

**AP_LOCAL**
q 源はローカル・ノードにあります。

**AP_DOMAIN**
q 源は U 加エンド・ノードに属しています。

**AP_CROSS_DOMAIN**
q 源はローカル・ノードのドメイン内にはありません。

**directory_lu_detail.entry_type**
ディレクトリー項目のタイプを X 定します。これは、以下のいずれかの値にすることができます。

**AP_HOME**
ローカル q 源。

**AP_CACHE**
キャッシュ項目。

**AP_REGISTER**
登録済みの q 源 (NN のみ)。

**directory_lu_detail.wild_card**
LU と一致するワイルドカードのタイプを X 定します。

**AP_OTHER**
認識されないタイプの LU 項目。

**AP_EXPLICIT**
全 **lu_name** が、この LU の位置決めに H 用されます。

**AP_PARTIAL_WILDCARD**
**lu_name** の非スペース t 分のみが、この LU の位置決めに H 用されます。

**AP_FULL_WILDCARD**
すべての **lu_names** がこの LU に送信されます。

**directory_lu_detail.apprent_lu_owner_name**
NN および BrNN のみ: 完全修飾見かけ所有者 CP 名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

見かけ LU 所有者 が実 LU 所有者の場合は、このフィールドは 2 進ゼロに設定されます。

見かけ LU 所有者 が実 LU 所有者でない場合は、このフィールドは見かけ LU 所有者の名前に設定されます。

q 源が BrNN のドメインで EN によって所有されていれば、 BrNN の NNS のディレクトリーでは、実 LU 所有者は見かけ LU 所有者ではありません。この場合、実 LU 所有者は EN ですが、見かけ所有者は BrNN です。

その他のノード・タイプ: このフィールドは 2 進ゼロに設定されます。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DIRECTORY_STATS

この verb は Communications Server にのみ適用されます。

QUERY_DIRECTORY_STATS は、ディレクトリー・データベース統計を戻します。(エンド・ノードの場合は、キャッシュ情報を 2 照する統計は予約済みです)。この verb をH用してネットワーク・ロケート・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、この情報をH用してディレクトリー・キャッシュのサイズを調整することができます。このサイズは、ノードの初期設定 ~ に構成可能です。

## VCB 構造体

```
typedef struct query_directory_stats
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned long   max_caches;          /* max number of cache entries    */
        unsigned long   cur_caches;          /* cache entry count              */
        unsigned long   cur_home_entries;    /* home entry count               */
        unsigned long   cur_reg_entries;     /* registered entry count         */
        unsigned long   cur_directory_entries;
                                             /* current number of dir entries  */
        unsigned long   cache_hits;          /* count of cache finds           */
        unsigned long   cache_misses;        /* count of resources found by    */
                                             /* broadcast search (not cache)   */
        unsigned long   in_locates;          /* locates in                     */
        unsigned long   in_bcast_locates;    /* broadcast locates in           */
        unsigned long   out_locates;         /* locates out                    */
        unsigned long   out_bcast_locates;   /* broadcast locates out          */
        unsigned long   not_found_locates;   /* unsuccessful locates           */
        unsigned long   not_found_bcast_locates;
                                             /* unsuccessful broadcast         */
                                             /* locates                        */
        unsigned long   locates_outstanding; /* total outstanding locates      */
        unsigned char   reserva[20];         /* reserved                       */
} QUERY_DIRECTORY_STATS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DIRECTORY_STATS

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**max_caches**
予約済み。

**cur_caches**
予約済み。

**cur_home_entries**
現在のホーム項目の数。

**cur_reg_entries**
現在の登録済み項目の数。

**cur_directory_entries**
現在ディレクトリーに入っている項目の合計数。

**cache_hits**
予約済み。

**cache_misses**
予約済み。

**in_locates**
受信した宛先ロケートの数。

**in_bcast_locates**
受信した同報通信ロケートの数。

**out_locates**
送信した宛先ロケートの数。

**out_bcast_locates**
送信した同報通信ロケートの数。

**not_found_locates**
“見つからない” というメッセージで戻された宛先ロケートの数。

**not_found_bcast_locates**
“見つからない” というメッセージで戻された同報通信ロケートの数。

**locates_outstanding**
未解決の宛先ロケートと同報通信ロケートの両方の現在の数。

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DLC

QUERY_DLC は、ノードで定義されている DLC に関する情報のリストを戻します。この情報は、『決定済みデータ』 (実行中に動的に収集されたデータ) および『定義済みデータ』 (DEFINE_DLC のアプリケーションによって提供されたデータ) として構造化されます。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の DLC に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **dlc_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、 **dlc_name** 順に配列されます。配列はまず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、 ASCII の辞書配列の順序で行われます ( 8 準の MIB 配列に準拠)。

AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された順序に従って次の項目から + O されます (X 定された項目が存在するかしないかに関係なく)。

## VCB 構造体

```
typedef struct query_dlc
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* ver attributes                */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   dlc_name[8];         /* name of DLC                   */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   state;               /* State of the DLC              */
        unsigned char   dlc_type;            /* DLC type                      */
} DLC_SUMMARY;

typedef struct dlc_detail
{
        unsigned short  overlay_size;        /* size of this 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
{
        DESCRIPTION     description;         /* resource description          */
        unsigned char   dlc_type;            /* DLC type                      */
        unsigned char   neg_ls_supp;         /* negotiable LS support         */
        unsigned char   port_types;          /* allowable port types          */
        unsigned char   retry_flags;         /* conditions for automatic      */
                                             /* retries                       */
        unsigned short  max_activaion_attempts;
                                             /* how many automatic retries?   */
        unsigned short  activation_delay_timer;
                                             /* delay between automatic       */
                                             /* retries                       */
        unsigned char    reserv3[6];         /* reserved                      */
        unsigned short  dlc_spec_data_len;   /* Length of DLC specific data   */
} DLC_DEF_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DLC

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

X定された **dlc_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O 点を X 定するために H 用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によって X 定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**dlc_name**
DLC 名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**dlc_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dlc_summary.dlc_name**
DLC 名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**dlc_summary.description**
q 源の説明 (DEFINE_DLC で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**dlc_summary.state**
DLC の状態。このフィールドは次のいずれかに設定されます。

AP_ACTIVE
AP_NOT_ACTIVE
AP_PENDING_INACTIVE

**dlc_summary.dlc_type**
DLC のタイプ。パーソナル・コミュニケーションズまたは Communications Serverは、以下のタイプをサポートします。

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

**dlc_detail.overlay_size**
この項目内のバイトの数 (dlc_spec_data を含む)。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dlc_detail.dlc_name**
DLC 名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**dlc_detail.det_data.state**
DLC の状態。このフィールドは次のいずれかに設定されます。

AP_ACTIVE
AP_NOT_ACTIVE
AP_PENDING_INACTIVE

**dlc_detail.def_data.description**
q 源の説明 (DEFINE_DLC で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**dlc_detail.def_data.dlc_type**
DLC のタイプ。パーソナル・コミュニケーションズまたは Communications Serverは、以下のタイプをサポートします。

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

**dlc_detail.def_data.neg_ls_supp**
DLC が折衝可能リンク・ステーションをサポートするかどうか (AP_YES または AP_NO) をX定します。

**dlc_detail.def_data.port_types**
提供された **dlc_type** の許容ポート・タイプをX定します。この値は、以下の 1 つの値またはいくつかの値を OR で結合した値と対応しています。

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

**dlc__detail.def_data.retry_flags**
AP_INHERIT_RETRY フラグが **def_data.retry_flags** の DEFINE_LS と DEFINE_PORT の両方に設定されている場合、このフィールドは、この DLC に定義されたリンク・ステーションが自動再 n 行される条件をX定します。これはビット・フィールドであり、以下の値をビット単位で OR で結合した任意の値を取ることができます。

**AP_RETRY_ON_START**
リンクの h 動化を n 行しているときにリモート・ノードから応答がないと、 h 動化が再 n 行されます。 h 動化を n 行しているときに基本ポートが非アクティブ状態であると、「プログラム」はそれを h 動化しようとします。

**AP_RETRY_ON_FAILURE**
リンクがアクティブまたは保留アクティブ状態のときに失敗すると、リンクの h 動化が再 n 行されます。 h 動化を n 行しているときに基本ポートが失敗すると、「プログラム」はそれを h 動化しようとします。

**AP_RETRY_ON_DISCONNECT**
リンクがリモート・ノードによって正常停 _ されると、リンクの h 動化が再 n 行されます。

**AP_DELAY_APPLICATION_RETRIES**
アプリケーションによって + O された (START_LS またはオンデマンド・リンク h 動化をH用して) リンク h 動化再 n 行は、**activation_delay_timer** をH用して歩調合わせされます。

**AP_INHERIT_RETRY**
このフラグは効果を生じません。

**dlc_detail.def_data.max_activation_attempts**
少なくとも 1 つのフラグが **def_data.retry_flags** の DEFINE_LS に設定され、 DEFINE_LS の **def_data.max_activation_attempts** が AP_USE_DEFAULTS に設定され、 DEFINE_PORT の **def_data.max_activation_attempts** が AP_USE_DEFAULTS に設定されている場合を除き、このフィールドは効果を生じません。

このフィールドは、リモート・ノードが無応答の場合、または基本ポートが非 h 動状態の場合に「プログラム」によって許容される再 n 行の回数をX定します。この回数には、自動再 n 行とアプリケーション主導型の h 動化 n 行の両方の回数も含まれます。

この限度に達すると、自動再 n 行はこれ以上行われません。この条件は、STOP_LS、STOP_PORT、STOP_DLC、または成功した h 動化によってリセットされます。 START_LS または OPEN_LU_SSCP_SEC_RQ によって 1 回の h 動化 n 行が行われますが、 h 動化に失敗すると、再 n 行は行われません。

ゼロは ’限度がない’ ことを意味します。AP_USE_DEFAULTS は ’限度がない’ ことを意味します。

**dlc_detail.def_data.activation_delay_timer**

少なくとも 1 つのフラグが **def_data.retry_flags** の DEFINE_LS に設定され、 DEFINE_LS の **def_data.max_activation_attempts** が AP_USE_DEFAULTS に設定され、 DEFINE_PORT の **def_data.max_activation_attempts** が AP_USE_DEFAULTS に設定されている場合を除き、このフィールドは効果を生じません。

このフィールドは、 AP_DELAY_APPLICATION_RETRIES ビットが **def_data.retry_flags** に設定されている場合に、「プログラム」が自動再 n 行間、およびアプリケーション主導型 h 動化 n 行間に待機する C 数を X 定します。

ゼロの値または AP_USE_DEFAULTS を X 定すると、デフォルトのタイマーが 30 C 間 H 用されます。

**dlc_detail.def_data.dlc_spec_data_len**

DLC のタイプに固有なデータの埋め込みなしの長さ (バイト数)。このデータは DLC_DETAIL 構造体と連結されます。このデータは、 4 バイト境&の終わりまで埋め込まれています。このフィールドは、常にゼロに設定しておかなければなりません。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_DLC_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DLUR_DEFAULTS

QUERY_DLUR_DEFAULTS をH用すれば、ユーザーは、 DEFINE_DLUR_DEFAULTS verb をH用して定義されたデフォルトを照会することができます。

## VCB 構造体

```
typedef struct query_dlur_defaults
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        DESCRIPTION     description;         /* resource description          */
        unsigned char   dlus_name[17];       /* DLUS name                     */
        unsigned char   bkup_dlus_name[17];  /* Backup DLUS name              */
        unsigned char   reserv3;             /* reserved                      */
        unsigned short  dlus_retry timeout;  /* DLUS Retry Timeout            */
        unsigned short  dlus_retry_limit;    /* DLUS Retry Limit              */
        unsigned char   reserv4[16];         /* reserved                      */
} QUERY_DLUR_LU;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DLUR_LU

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

**description**

ｑ源の説明。このフィールドの長さは 4 バイトの倍数で、非ゼロでなければなりません。

**dlus_name**

デフォルトとして処理される DLUS ノードの名前。この値は、すべてゼロに設定されるか、あるいは、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文ｚ列からなる 17 バイトの文ｚ列に設定され、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**bkup_dlus_name**

バックアップ・デフォルトとして処理される DLUS ノードの名前。この値はすべてゼロに設定されるか、あるいは、1 つのドットで連結された 2 つのタ

イプ A の EBCDIC 文 z 列からなる 17 バイトの文 z 列に設定され、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) LU の位置。戻される唯一の値は、次のとおりです。

**dlus_retry_timeout**
DLUS にコンタクトするための、2 番目の n 行とその後の n 行との間の間V (C 数)。最初の n みと最初の再 n 行の間の間V は、常に 1 C です。

**dlus_retry_limit**
DLUS への接続が最初に失敗したとき以降に行われる再 n 行の最大数。X'FFFF' をX定すると、「プログラム」は無限に再 n 行を繰り返します。

関係のある START_NODE パラメーターが設定されていないためにこの verb が実行されない場合は、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ノードがまだ + O されていないために verb が実行されない場合には、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_INVALID_VERB

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

STOP_NODE verb が発行されたためにこの verb が実行されない場合は、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DLUR_LU

QUERY_DLUR_LU は、DLUR サポート LU に関する情報のリストを戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の LU に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **lu_name** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**lu_name** 順に配列されます。配列はまず名前の長さ順に行われ、次に、名前の長さが同じ場合は、 ASCII の辞書配列の順序で行われます ( 8 準の MIB 配列に準拠)。

AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

戻された LU のリストは、 **pu_name** 別にフィルター処理することもできるし、 LU がローカルであるかダウンストリームであるか、あるいはその両方であるかによってフィルター処理することもできます。 PU 別のフィルター処理を行いたい場合は、**pu_name** フィールドを設定する , 要があります (そうしない場合は、このフィールドをすべてゼロに設定する , 要があります)。位置別にフィルター処理を行いたい場合は、 **filter** フィールドを AP_INTERNAL か AP_DOWNSTREAM に設定する , 要があります (そうしない場合は、このフィールドを AP_NONE に設定する , 要があります)。

## VCB 構造体

```
typedef struct query_dlur_lu
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  total_num_entries;   /* total number of entries        */
        unsigned char   list_options;        /* listing options                */
        unsigned char   reserv3;             /* reserved                       */
        unsigned char   lu_name[8];          /* name of LU                     */
        unsigned char   pu_name[8];          /* name of PU to filter on        */
        unsigned char   filter;              /* reserved                       */
} QUERY_DLUR_LU;

typedef struct dlur_lu_summary
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   lu_name[8];          /* name of LU                     */
} DLUR_LU_SUMMARY;

typedef struct dlur_lu_detail
{
        unsigned short  overlay_size;      /* size of this entry          */
```

```
        unsigned char   lu_name[8];        /* name of LU                  */
        unsigned char   pu_name[8];        /* 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;
```

## 指定Ｑｉａ一？一

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DLUR_LU

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

X定された **lu_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から + O されます。

**lu_name**
照会される LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**pu_name**
PU 名フィルター。このフィールドをすべてゼロに設定するか、 8 バイト英数 z のタイプ A の EBCDIC 文 z 列に設定し (文 z で O まる)、右側に EBCDIC スペースを埋め込む , 要があります。このフィールドを設定すると、X 定の PU と関連がある LU だけが戻されます。すべてゼロに設定すると、このフィールドは無 k されます。

**filter** 位置フィルター。戻された LU に対して位置別のフィルター処理を行うかどうかを X 定します (AP_INTERNAL または AP_DOWNSTREAM)。フィルター処理が , 要でない場合は、このフィールドを AP_NONE に設定する , 要があります。

## 戻り Q i a ー ? ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**dlur_lu_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dlur_lu_summary.lu_name**
LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。

**dlur_lu_detail.overlay_size**
この項目内の番号の数 (U 加 RSCV を含む)。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dlur_lu_detail.lu_name**
LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。

**dlur_lu_detail.pu_name**
LU と関連した PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**dlur_lu_detail.dlus_name**
SSCP-LU セッションがアクティブの場合の DLUS ノードの名前。この名前は、1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなる 17 バイトの文 z 列で、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) SSCP-LU セッションがアクティブでなければ、このフィールドはすべてゼロに設定されます。

**dlur_lu_detail.lu_location**
LU の位置。戻される唯一の値は、次のとおりです。

AP_INTERNAL
AP_DOWNSTREAM

**dlur_lu_detail.nau_address**
LU のネットワーク・アドレス単位アドレス。範囲は、1 〜 255 です。

**dlur_lu_detail.plu_name**
LU がアクティブ PLU-SLU セッションをもっている場合の PLU の名前。この名前は、1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなる 17 バイトの文 z 列で、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) PLU-SLU セッションがアクティブでなければ、このフィールドはすべてゼロに設定されます。

**dlur_lu_detail.rscv_len**
この値は、常にゼロになります。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_INVALID_FILTER_OPTION
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DLUR_PU

QUERY_DLUR_PU は、DLUR サポート PU に関する情報のリストを戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の PU に関する情報または いくつかの 『固まり』 に分けられたリスト情報を入手するには、 **pu_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**pu_name** 順に配列されます。配列はまず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、 ASCII の辞書配列の順序で行われます ( 8 準の MIB 配列に準拠)。

AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

戻された PU のリストは、 **dlus_name** 別にフィルター処理することもできるし、 PU がローカルであるか、ダウンストリームであるか、あるいはその両方であるかによってフィルター処理することもできます。DLUS 別にフィルター処理を行いたい場合は、 **dlus_name** フィールドを設定する , 要があります (そうしない場合は、このフィールドをすべてゼロに設定する , 要があります)。PU 位置別のフィルター処理を行いたい場合は、 **filter** フィールドを AP_INTERNAL または AP_DOWNSTREAM に設定する , 要があります (フィルター処理が , 要でない場合は、このフィールドを AP_NONE に設定する , 要があります)。

## VCB 構造体

```
typedef struct query_dlur_pu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  total_num_entries;   /* total number of entries       */
        unsigned char   list_options;        /* listing options               */
        unsigned char   reserv3;             /* reserved                      */
        unsigned char   pu_name[8];          /* name of PU                    */
        unsigned char   dlus_name[17];       /* fully qualified DLUS name     */
        unsigned char   filter;              /* local/downstream filter       */
} QUERY_DLUR_PU;

typedef struct dlur_pu_summary
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   pu_name[8];          /* name of PU                    */
        unsigned char   description[RD_LEN]; /* resource description          */
} DLUR_PU_SUMMARY;

typedef struct dlur_pu_detail
{
        unsigned short  overlay_size;        /* size of this entry            */
```

```
        unsigned char   pu_name[8];          /* name of PU                    */
        unsigned char   description[RD_LEN]; /* resource description          */
        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           */
        unsigned short  dlus_retry_timeout;  /* DLUS retry timeout            */
        unsigned short  dlus_retry_limit;    /* DLUS retry limit              */
} DLUR_PU_DETAIL;

typedef struct fqpcid
{
        unsigned char   pcid[8];             /* proc correlator identifier    */
        unsigned char   fqcp_name[17];       /* originator's network          */
                                             /* qualified CP name             */
        unsigned char   reserve3[3];         /* reserved                      */
} FQPCID;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DLUR_PU

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

X定された **pu_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O 点を X 定するために H 用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**pu_name**
照会される PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**dlus_name**
DLUS フィルター。このフィールドは、すべてゼロに設定するか、あるいは 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなる 17 バイトの文 z 列に設定し、右側に EBCDIC スペースを埋め込む , 要があります。このフィールドを設定すると、X定された DLUS ノードに対する SSCP-PU セッションと関連する PU だけが戻されます。すべてゼロに設定すると、このフィールドは無 k されます。

**filter** このフィールドは AP_NONE に設定する , 要があります。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**dlur_pu_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dlur_pu_summary.pu_name**
PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。

**dlur_pu_summary.description**
q 源の説明 (DEFINE_INTERNAL_PU でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**dlur_pu_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dlur_pu_detail.pu_name**
PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。

**dlur_pu_detail.description**
q 源の説明 (DEFINE_INTERNAL_PU でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**dlur_pu_detail.defined_dlus_name**
DEFINE_INTERNAL_PU verb または DEFINE_LS verb のいずれかで定義された DLUS ノードの名前 (**dspu_services** を AP_DLUR に設定)。この名前は、1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなる 17 バイトの文 z 列で、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**dlur_pu_detail.bkup_dlus_name**
DEFINE_INTERNAL_PU verb または DEFINE_LS verb のいずれかで定義されたバックアップ DLUS ノードの名前 (**dspu_services** を AP_DLUR に設定)。この名前は、1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなる 17 バイトの文 z 列で、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**dlur_pu_detail.pu_id**
DEFINE_INTERNAL_PU verb に定義された PU ID またはダウンストリーム PU から XID で入手された PU ID。これは、4 バイトの16 進数文 z 列です。ビット 0 ～ 11 はブロック番号に設定され、ビット 12 ～ 31 は PU を一意的に識別する ID 番号に設定されます。

**dlur_pu_detail.pu_location**
PU の位置。戻される唯一の値は、次のとおりです。

AP_INTERNAL
AP_DOWNSTREAM

**dlur_pu_detail.active_dlus_name**
PU が現在H用している DLUS ノードの名前。これは、1 つの EBCDIC ドットで連結された 2 つのタイプ A のEBCDIC 文 z 列からなる 17 バイトの文 z 列で、右側にEBCDIC スペースが埋め込まれています。 (それぞれの名前

は、埋め込みスペースのない最大 8 バイトの長さになります。) SSCP-PU セッションがアクティブでない場合は、このフィールドはすべてゼロに設定されます。

**dlur_pu_detail.ans_support**
自動ネットワーク・シャットダウン・サポート。 SSCP-LU セッションが非アクティブの場合は、このフィールドは予約済みです。このサポート設定は、SSCP-PU h 動化 ~ に DLUS から DLUR へ送信されます。このフィールドは、サブエリア・ノードが、 PU を制御する SSCP について自動ネットワーク・シャットダウン・プロシージャーを + O する場合に、リンク・レベル・コンタクトを継続するかどうかを X 定します。これは、以下のいずれかの値にすることができます。

AP_CONT
AP_STOP

**dlur_pu_detail.pu_status**
PU の状況 (DLUR から N 認される)。これは以下のいずれかの値に設定されます。

**AP_RESET**
PU はリセット状態になっています。

**AP_PEND_ACTPU**
PU はホストからの ACTPU を待っています。

**AP_PEND_ACTPU_RSP**
DLUR は、ACTPU を PU に転送した後、PU がそれに応答するのを待っています。

**AP_ACTIVE**
PU はアクティブ状態です。

**AP_PEND_DACTPU_RSP**
DLUR は、DACTPU を PU に転送した後、PU がそれに応答するのを待っています。

**AP_PEND_INOP**
DLUR は、PU を非 h 動化する前に、 , 要なすべてのイベントが完了するのを待っています。

**dlur_pu_detail.dlus_session_status**
現在 PU によって H 用されている DLUS パイプの状況。これは、以下のいずれかの値にすることができます。

AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE
AP_INACTIVE

**dlur_pu_detail.fqpcid.pcid**
パイプで H 用されたプロシージャー相関係数 ID。これは 8 バイトの 16 進文 z 列です。SSCP-PU セッションがアクティブ状態でなければ、このフィールドはゼロに設定されます。

# QUERY_DLUS

QUERY_DLUS は、DLUR によって認識された DLUS ノードに関する情報のリストを戻します。

この情報はリストとして戻されます。特定の DLUS ノードに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **dlus_name** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**dlus_name** 順に配列されます。配列はまず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます ( 8 準の MIB 配列に準拠)。

AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

この verb がパイプ統計を戻すことに注意してください。

## VCB 構造体

```
typedef struct query_dlus
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this 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        */
        unsigned short  num_active_pus;      /* num of active PUs using pipe  */
        PIPE_STATS      pipe_stats;          /* pipe statistics               */
} DLUS_DATA;

typedef struct pipe_stats
{
        unsigned long   reqactpu_sent;       /* REQACTPUs sent to DLUS        */
        unsigned long   reqactpu_rsp_received;
                                             /* RSP(REQACTPU)s received       */
                                             /* from DLUS                     */
        unsigned long   actpu_received;      /* ACTPUs received from DLUS     */
        unsigned long   actpu_rsp_sent;      /* RSP(ACTPU)s sent to DLUS      */
        unsigned long   reqdactpu_sent;      /* REQDACTPUs sent to DLUS       */
```

```
        unsigned long   reqdactpu_rsp_received;
                                        /* RSP(REQDACTPU)s received       */
                                        /* from DLUS                      */
        unsigned long   dactpu_received;    /* DACTPUs received from DLUS     */
        unsigned long   dactpu_rsp_sent;    /* RSP(DACTPU)s sent to DLUS      */
        unsigned long   actlu_received;     /* ACTLUs received from DLUS      */
        unsigned long   actlu_rsp_sent;     /* RSP(ACTLU)s sent to DLUS       */
        unsigned long   dactlu_received;    /* DACTLUs received from DLUS     */
        unsigned long   dactlu_rsp_sent;    /* RSP(DACTLU)s sent to DLUS      */
        unsigned long   sscp_pu_mus_rcvd;   /* MUs for SSCP-PU                */
                                        /* sessions received              */
        unsigned long   sscp_pu_mus_sent;   /* MUs for SSCP-PU sessions sent  */
        unsigned long   sscp_lu_mus_rcvd;   /* MUs for SSCP-LU sessions       */
                                        /* received                       */
        unsigned long   sscp_lu_mus_sent;   /* MUs for SSCP-LU sessions sent  */
} PIPE_STATS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DLUS

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

X定された **dlus_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**dlus_name**
照会される DLUS の名前。このフィールドは、すべてゼロに設定するか、あるいは 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなる 17 バイトの文 z 列に設定し、右側に EBCDIC スペースを埋め込む , 要があります。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**dlus_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dlus_data.dlus_name**
DLUS の名前。この名前は、1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなる 17 バイトの文 z 列で、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**dlus_data.is_default**
DLUS ノードが、DEFINE_DLUR_DEFAULTS verb によってデフォルトとしてX定されたかどうか (AP_YES または AP_NO) をX定します。

**dlus_data.is_backup_default**
DLUS ノードが、DEFINE_DLUR_DEFAULTS verb によってバックアップ・デフォルトとしてX定されたかどうか (AP_YES または AP_NO) をX定します。

**dlus_data.pipe_state**
DLUS へのパイプの状態。以下のいずれかの値になります。

AP_ACTIVE
AP_PENDING_ACTIVE
AP_INACTIVE
AP_PENDING_INACTIVE

**dlus_data.num_active_pus**
現在 DLUS へのパイプをH用している PU の数。

**dlus_data.pipe_stats.reqactpu_sent**
パイプを介して DLUS に送信した REQACTPU の数。

**dlus_data.pipe_stats.reqactpu_rsp_received**
パイプを介して DLUS から受信した RSP(REQACTPU) の数。

**dlus_data.pipe_stats.actpu_received**
パイプを介して DLUS から受信した ACTPU の数。

**dlus_data.pipe_stats.actpu_rsp_sent**
パイプを介して DLUS に送信した RSP(ACTPU) の数。

**dlus_data.pipe_stats.reqdactpu_sent**
パイプを介して DLUS に送信した REQDACTPU の数。

**dlus_data.pipe_stats.reqdactpu_rsp_received**
パイプを介して DLUS から受信した RSP(REQDACTPU) の数。

**dlus_data.pipe_stats.dactpu_received**
パイプを介して DLUS から受信した DACTPU の数。

**dlus_data.pipe_stats.dactpu_rsp_sent**
パイプを介して DLUS に送信した RSP(DACTPU) の数。

**dlus_data.pipe_stats.actlu_received**
パイプを介して DLUS から受信した ACTLU の数。

**dlus_data.pipe_stats.actlu_rsp_sent**
パイプを介して DLUS に送信した RSP(ACTLU) の数。

**dlus_data.pipe_stats.dactlu_received**
パイプを介して DLUS から受信した DACTLU の数。

**dlus_data.pipe_stats.dactlu_rsp_sent**
パイプを介して DLUS に送信した RSP(DACTLU) の数。

**dlus_data.pipe_stats.sscp_pu_mus_rcvd**
パイプを介して DLUS から受信した SSCP-PU MU の数。

**dlus_data.pipe_stats.sscp_pu_mus_sent**
パイプを介して DLUS に送信した SSCP-PU MU の数。

**dlus_data.pipe_stats.sscp_lu_mus_rcvd**
パイプを介して DLUS から受信した SSCP-LU MU の数。

**dlus_data.pipe_stats.sscp_lu_mus_sent**
パイプを介して DLUS に送信した SSCP-LU MU の数。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DLUS_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DOWNSTREAM_LU

この verb は Communications Server にのみ適用されます。

QUERY_DOWNSTREAM_LU は、 DLUR 集信または PU 集信、あるいはその両方からサービスを受けるダウンストリーム LU に関する情報を戻します。この情報は、決定済みデータ (実行 ~ に動的に収集されたデータ) および定義済みデータとして構成されます。 (定義済みデータは、アプリケーションによって DEFINE_DOWNSTREAM_LU verb で提供されます。ただし、DLUR サポート LU の場合、ダウンストリーム LU が h 動化されると、暗黙定義データが H 用される点に注意してください。)

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のローカル LU に関する情報またはいくつかの固まりに分けられたリスト情報を入手するには、 **dslu_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。

戻された LU は、ローカル・ノードによって提供されるサービスのタイプ別、または LU 関連ダウンストリーム PU 別、あるいはその両方でフィルター処理することができます。サービス・タイプ別のフィルター処理を行いたい場合は、 **dspu_services** フィールドを AP_PU_CONCENTRATION または AP_DLUR に設定する , 要があります (それ以 0 の場合は、このフィールドを AP_NONE に設定する , 要があります)。PU 別のフィルター処理を行いたい場合は、 **dspu_name** フィールドを設定する , 要があります (それ以 0 の場合は、このフィールドをすべてゼロに設定する , 要があります)。

## VCB 構造体

```
typedef struct query_downstream_lu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* Verb attributes               */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  total_num_entries;   /* total number of entries       */
        unsigned char   list_options;        /* listing options               */
        unsigned char   reserv3;             /* reserved                      */
        unsigned char   dslu_name[8];        /* Downstream LU name            */
        unsigned char   dspu_name[8];        /* Downstream PU name filter     */
        unsigned char   dspu_services;       /* filter on DSPU services type  */
} QUERY_DOWNSTREAM_LU;

typedef struct downstream_lu_summary
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   dslu_name[8];        /* LU name                       */
        unsigned char   dspu_name[8];        /* PU name                       */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   dspu_services;       /* type of service provided to   */
                                             /* downstream node               */
```

```
        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
{
        unsigned short  overlay_size;        /* size of this 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 services 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 sess stats    */
        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[RD_LEN]; /* resource description           */
        unsigned char   nau_address;         /* NAU address                    */
        unsigned char   dspu_name[8];        /* Downstream PU name             */
        unsigned char   host_pu_name;        /* host LU or pool name           */
        unsigned char   allow_timeout;       /* Allow timeout of host LU?      */
        unsigned char   delayed_logon;       /* Allow delayed logo to host LU  */
        unsigned char   reserv2[6];          /* reserved                       */
} DOWNSTREAM_LU_DEF_DATA

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size        */
        unsigned short  send_ru_size;        /* session send RU size           */
        unsigned short  max_send_btu_size;   /* max send BTU size              */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size               */
        unsigned short  max_send_pac_win;    /* max send pacing win size       */
        unsigned short  cur_send_pac_win;    /* current send pacing win size   */
        unsigned short  max_rcv_pac_win;     /* max receive pacing win size    */
        unsigned short  cur_rcv_pac_win;     /* current receive pacing         */
                                             /* window size                    */
        unsigned long   send_data_frames;    /* number of data frames sent     */
        unsigned long   send_fmd_data_frames;
                                             /* num of FMD data frames sent    */
        unsigned long   send_data_bytes;     /* number of data bytes sent      */
        unsigned long   rcv_data_frames;     /* num data frames received       */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames recvd   */
        unsigned long   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[8];          /* Link station name              */
        unsigned char   pacing type;         /* type of pacing in use          */
} SESSION_STATS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DOWNSTREAM_LU

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをXすポインター。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

X定された **dslu_name** (以下のパラメーターを 2 照）は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から + O されます。

**dslu_name**

照会されるローカル LU の名前。これは、8 バイト英数 z のタイプ A の

EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**dspu_name**
PU 名フィルター。このフィールドをすべてゼロに設定するか、 8 バイト英数 z のタイプ A の EBCDIC 文 z 列に設定し (文 z でOまる)、右側に EBCDIC スペースを埋め込む , 要があります。このフィールドを設定すると、X定された PU と関連する LU のみが戻されます。すべてゼロに設定すると、このフィールドは無 k されます。

**dspu_services**
DSPU サービス・フィルター。AP_PU_CONCENTRATION に設定すると、 PU 集信のサービスを受けるダウンストリーム LU のみが戻されます。AP_DLUR に設定すると、 DLUR サポート LU のみが戻されます。それ以 0 の場合、AP_NONE に設定すると、すべてのダウンストリーム LU に関する情報が戻されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**downstream_lu_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**downstream_lu_summary.dslu_name**
照会されるローカル LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**downstream_lu_summary.dspu_name**
この LU がH用するローカル PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**downstream_lu_summary.description**
q 源の説明 (DEFINE_DOWNSTREAM_LU または DEFINE_DOWNSTREAM_LU_RANGE でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**downstream_lu_summary.dspu_services**
ローカル・ノードがリンクを介してダウンストリーム LU に提供するサービスをX定します。これは、以下のいずれかに設定されます。

**AP_PU_CONCENTRATION**
ダウンストリーム LU に PU 集信を提供するローカル・ノード。

**AP_DLUR**
ダウンストリーム LU に DLUR 集信を提供するローカル・ノード。

**downstream_lu_summary.nau_address**
LU のネットワーク・アドレス単位アドレス。範囲は 1 ～ 255。

**downstream_lu_summary.lu_sscp_sess_active**
LU-SSCP セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

**downstream_lu_summary.plu_sess_active**
PLU-SLU セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

**downstream_lu_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**downstream_lu_detail.dslu_name**
照会されるローカル LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。

**downstream_lu_detail.det_data.lu_sscp_sess_active**
ダウンストリーム LU に対する LU-SSCP セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

**downstream_lu_detail.det_data.plu_sess_active**
ダウンストリーム LU に対する PLU-SLU セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

**downstream_lu_detail.det_data.dspu_services**
ローカル・ノードがリンクを介してダウンストリーム LU に提供するサービスをX定します。以下のいずれかの値に設定されます。

**AP_PU_CONCENTRATION**
ダウンストリーム LU に PU 集信を提供するローカル・ノード。

**AP_DLUR**
ダウンストリーム LU に DLUR 集信を提供するローカル・ノード。

**downstream_lu_detail.det_data.lu_sscp_stats.rcv_ru_size**
受信 RU の最大サイズ。**downstream_lu_detail.det_data.dspu_services** を AP_PU_CONCENTRATION に設定した場合は、このフィールドは予約済みです。

**downstream_lu_detail.det_data.lu_sscp_stats.send_ru_size**
送信 RU の最大サイズ。**downstream_lu_detail.det_data.dspu_services** が AP_PU_CONCENTRATION に設定されていれば、このフィールドは予約済みです。

**downstream_lu_detail.det_data.lu_sscp_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**downstream_lu_detail.det_data.lu_sscp_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**downstream_lu_detail.det_data.lu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**downstream_lu_detail.det_data.lu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**downstream_lu_detail.det_data.lu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**downstream_lu_detail.det_data.lu_sscp_stats.cur_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**downstream_lu_detail.det_data.lu_sscp_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**downstream_lu_detail.det_data.lu_sscp_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**downstream_lu_detail.det_data.lu_sscp_stats.send_data_bytes**
送信された通常フロー・データ・バイト数。

**downstream_lu_detail.det_data.lu_sscp_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**downstream_lu_detail.det_data.lu_sscp_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**downstream_lu_detail.det_data.lu_sscp_stats.rcv_data_bytes**
受信された通常フロー・データ・バイトの数。

**downstream_lu_detail.det_data.lu_sscp_stats.sidh**
セッション ID 上位バイト。

**downstream_lu_detail.det_data.lu_sscp_stats.sidl**
セッション ID の下位バイト。

**downstream_lu_detail.det_data.lu_sscp_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、BIND の送信側はこのフィールドをゼロに設定し、 BIND の送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドを 1 に設定します。

**downstream_lu_detail.det_data.lu_sscp_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**downstream_lu_detail.det_data.ds_plu_stats.rcv_ru_size**
受信 RU の最大サイズ。

**downstream_lu_detail.det_data.ds_plu_stats.send_ru_size**
送信 RU の最大サイズ。

**downstream_lu_detail.det_data.ds_plu_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**downstream_lu_detail.det_data.ds_plu_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**downstream_lu_detail.det_data.ds_plu_stats.max_send_pac_win**
このセッションでの送信ペーシング・ウィンドウの最大サイズ。

**downstream_lu_detail.det_data.ds_plu_stats.cur_send_pac_win**
このセッションでの送信ペーシング・ウィンドウの現行サイズ。

**downstream_lu_detail.det_data.ds_plu_stats.max_rcv_pac_win**
このセッションでの受信ペーシング・ウィンドウの最大サイズ。

**downstream_lu_detail.det_data.ds_plu_stats.cur_rcv_pac_win**
このセッションでの受信ペーシング・ウィンドウの現行サイズ。

**downstream_lu_detail.det_data.ds_plu_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**downstream_lu_detail.det_data.ds_plu_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**downstream_lu_detail.det_data.ds_plu_stats.send_data_bytes**
送信された通常フロー・データ・バイトの数。

**downstream_lu_detail.det_data.ds_plu_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**downstream_lu_detail.det_data.ds_plu_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**downstream_lu_detail.det_data.ds_plu_stats.rcv_data_bytes**
受信された通常フロー・データ・バイトの数。

**downstream_lu_detail.det_data.ds_plu_stats.sidh**
セッション ID 上位バイト。

**downstream_lu_detail.det_data.ds_plu_stats.sidl**
セッション ID 下位バイト。

**downstream_lu_detail.det_data.ds_plu_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、BIND の送信側はこのフィールドをゼロに設定し、 BIND の送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドを 1 に設定します。

**downstream_lu_detail.det_data.ds_plu_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**downstream_lu_detail.det_data.plu_stats.pacing_type**
ダウンストリーム PLU-SLU セッションで H 用される受信ペーシング・タイプ。このフィールドには、AP_NONE または AP_PACING_FIXED 値を入れることができます。

**downstream_lu_detail.det_data.lu_sscp_pacing_type**
LU-SSCP セッションでH用される受信ペーシング。このフィールドには、AP_NONE 値を入れることができます。

**downstream_lu_detail.det_data.us_plu_stats.send_ru_size**
送信 RU の最大サイズ。

**downstream_lu_detail.det_data.us_plu_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**downstream_lu_detail.det_data.us_plu_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**downstream_lu_detail.det_data.us_plu_stats.max_send_pac_win**
このセッションでの送信ペーシング・ウィンドウの最大サイズ。

**downstream_lu_detail.det_data.us_plu_stats.cur_send_pac_win**
このセッションでの送信ペーシング・ウィンドウの現行サイズ。

**downstream_lu_detail.det_data.us_plu_stats.max_rcv_pac_win**
このセッションでの受信ペーシング・ウィンドウの最大サイズ。

**downstream_lu_detail.det_data.us_plu_stats.cur_rcv_pac_win**
このセッションでの受信ペーシング・ウィンドウの現行サイズ。

**downstream_lu_detail.det_data.us_plu_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**downstream_lu_detail.det_data.us_plu_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**downstream_lu_detail.det_data.us_plu_stats.send_data_bytes**
送信された通常フロー・データ・バイトの数。

**downstream_lu_detail.det_data.us_plu_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**downstream_lu_detail.det_data.us_plu_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**downstream_lu_detail.det_data.us_plu_stats.rcv_data_bytes**
受信された通常フロー・データ・バイトの数。

**downstream_lu_detail.det_data.us_plu_stats.sidh**
セッション ID 上位バイト。
**downstream_lu_detail.det_data_.dspu_services** が AP_PU_CONCENTRATION に設定されていれば、このフィールドは予約済みです。

**downstream_lu_detail.det_data.us_plu_stats.sidl**
セッション ID 下位バイト。
**downstream_lu_detail.det_data_.dspu_services** が AP_PU_CONCENTRATION に設定されていれば、このフィールドは予約済みです。

**downstream_lu_detail.det_data.us_plu_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、BIND の送信側はこのフィールドをゼロ

に設定し、 BIND の送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドを 1 に設定します。
**downstream_lu_detail.det_data_.dspu_services** が AP_PU_CONCENTRATION に設定されていれば、このフィールドは予約済みです。

**downstream_lu_detail.det_data.us_plu_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。
**downstream_lu_detail.det_data_.dspu_services** が AP_PU_CONCENTRATION に設定されていれば、このフィールドは予約済みです。

**downstream_lu_detail.det_data.us_plu_stats.pacing_type**
アップストリーム PLU-SLU セッションでH用される受信ペーシング・タイプ。このフィールドには、AP_NONE または AP_PACING_FIXED 値を入れることができます。

**downstream_lu_detail.det_data.host_lu_name**
ダウンストリーム LU のマップ先のホスト LU の名前、または PLU-SLU セッションが最後にアクティブになっていたときにマップされた先のホスト LU の名前。この名前は、**def_data.host_lu_name** とは異なることがあります。それは、この名前がホスト LU プールの名前である場合があるからです。

**downstream_lu_detail.det_data.host_pu_name**
ダウンストリーム PU のマップ先のホスト PU の名前、または PLU-SLU セッションが最後にアクティブになっていたときにマップされた先のホスト LU の名前。

**downstream_lu_detail.def_data.description**
q 源の説明 (DEFINE_DOWNSTREAM_LU または DEFINE_DOWNSTREAM_LU_RANGE でX定)。

**downstream_lu_detail.def_data.nau_address**
LU のネットワーク・アドレス単位アドレス。範囲は 1 〜 255。

**downstream_lu_detail.def_data.dspu_name**
LU と関連する PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**downstream_lu_detail.def_data.host_lu_name**
ダウンストリーム LU がマップされる先のホスト LU またはホスト LU プールの名前。 LUの場合、これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、見側に EBCDIC スペースが埋め込まれています。LU プールの場合、パーソナル・コミュニケーションズまたは Communications Serverはこのフィールドに文 z セットをX定しません。このフィールドは、DLUR のサービスを受けるダウンストリーム LU 用に予約されています。

**downstream_lu_detail.def_data.allow_timeout**
セッションが、ホスト LU 定義にX定された**タイムアウト**~間のあいだ非アクティブになっている場合に、パーソナル・コミュニケーションズまたは Communications Serverが、このダウンストリーム LU によってH用されるホスト LU をタイムアウトにすることが許可されているかどうかをX定します (AP_YES または AP_NO)。

**downstream_lu_detail.def_data.delayed_logon**
ダウンストリーム LU から最初のデータを受信するまで、パーソナル・コミュニケーションズまたは Communications Serverがダウンストリーム LU とホスト LU との接続を遅らせるかどうかをX定します。その代わり、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 (AP_YES または AP_NO)。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DOWNSTREAM_PU

この verb は Communications Server にのみ適用されます。

QUERY_DOWNSTREAM_PU は、ダウンストリーム PU に関する情報を戻します (DEFINE_LS verb で定義)。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のローカル PU に関する情報またはいくつかの固まりに分けられたリスト情報を入手するには、 **dspu_name** フィールドを設定する , 要があります。そうでなければ ( **list_options** フィールドに AP_FIRST_IN_LIST を設定する場合)、このフィールドは無 k されます。

PU のリストは、ローカル・ノードがダウンストリーム PU に提供するサービスのタイプ別にフィルター処理することができます。これを行うには、 **dspu_services** フィールドを AP_PU_CONCENTRATION または AP_DLUR に設定する , 要があります。

## VCB 構造体

```
typedef struct query_downstream_pu
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   attributes;          /* Verb attributes                */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  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             */
        unsigned char   dspu_services;       /* filter on DSPU services type   */
} QUERY_DOWNSTREAM_PU;

typedef struct downstream_pu_data
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   dspu_name[8];        /* PU name                        */
        unsigned char   description[RD_LEN]; /* resource description           */
        unsigned char   ls_name[8];          /* Link name                      */
        unsigned char   pu_sscp_sess_active; /* Is PU-SSCP session active      */
        unsigned char   dspu_services;       /* DSPU service type              */
        SESSION _STATS  pu_sscp_stats;       /* SSCP-PU session stats          */
        unsigned char   reserva[20];         /* reserved                       */
} DOWNSTREAM_PU_DATA

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size        */
        unsigned short  send_ru_size;        /* session send RU size           */
        unsigned short  max_send_btu_size;   /* max send BTU size              */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size               */
        unsigned short  max_send_pac_win;    /* max send pacing win size       */
        unsigned short  cur_send_pac_win;    /* current send pacing win size   */
        unsigned short  max_rcv_pac_win;     /* max receive pacing win size    */
```

```
        unsigned short  cur_rcv_pac_win;     /* current receive pacing         */
                                             /* window size                    */
        unsigned long   send_data_frames;    /* number of data frames sent     */
        unsigned long   send_fmd_data_frames;
                                             /* num of FMD data frames sent    */
        unsigned long   send_data_bytes;     /* number of data bytes sent      */
        unsigned long   rcv_data_frames;     /* num data frames received       */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames recvd   */
        unsigned long   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[8];          /* Link station name              */
        unsigned char   pacing_type;         /* type of pacing in use          */
} SESSION_STATS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DOWNSTREAM_PU

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをXすポインター。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

X定された **dslu_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O 点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**dspu_name**
照会されるダウンストリーム PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**dspu_services**
DSPU サービス・フィルター。AP_PU_CONCENTRATION に設定すると、 PU 集信のサービスを受けるダウンストリーム LU のみが戻されます。AP_DLUR に設定すると、 DLUR サポート LU のみが戻されます。それ以 0 の場合、AP_NONE に設定すると、すべてのダウンストリーム LU に関する情報が戻されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**downstream_pu_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**downstream_pu_data.dspu_name**
ダウンストリーム PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。

## QUERY_DOWNSTREAM_PU

**downstream_pu_data.description**
q 源の説明 (DEFINE_LS でX定)。

**downstream_pu_data.ls_name**
リンク・ステーションの名前。これは、ローカル= 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**downstream_pu_data.pu_sscp_sess_active**
ダウンストリーム PU との PU_SSCP セッションがアクティブかどうかを示します。 AP_YES または AP_NO のいずれかに設定されます。

**downstream_pu_data.dspu_services**
ローカル・ノードがリンクを介してダウンストリーム PU に提供するサービスをX定します。以下のいずれかの値に設定されます。

**AP_PU_CONCENTRATION**
ダウンストリーム LU に PU 集信を提供するローカル・ノード。

**AP_DLUR**
ダウンストリーム LU に DLUR 集信を提供するローカル・ノード。

**downstream_pu_data.pu_sscp_stats.rcv_ru_size**
受信 RU の最大サイズ。**downstream_lu_detail.det_data.dspu_services** が AP_PU_CONCENTRATION に設定されていれば、このフィールドは予約済みです。

**downstream_pu_data.pu_sscp_stats.send_ru_size**
送信 RU の最大サイズ。**downstream_lu_detail.det_data.dspu_services** が AP_PU_CONCENTRATION に設定されていれば、このフィールドは予約済みです。

**downstream_pu_data.pu_sscp_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**downstream_pu_data.pu_sscp_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**downstream_pu_data.pu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**downstream_pu_data.pu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**downstream_pu_data.pu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**downstream_pu_data.pu_sscp_stats.cur_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**downstream_pu_data.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。

**downstream_pu_data.pu_sscp_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**downstream_pu_data.pu_sscp_stats.send_data_bytes**
送信された通常フロー・データ・バイトの数。

**downstream_pu_data.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。

**downstream_pu_data.pu_sscp_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**downstream_pu_data.pu_sscp_stats.rcv_data_bytes**
受信された通常フロー・データ・バイトの数。

**downstream_pu_data.pu_sscp_stats.sidh**
セッション ID 上位バイト。

**downstream_pu_data.pu_sscp_stats.sidl**
セッション ID 下位バイト。

**downstream_pu_data.pu_sscp_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、BIND の送信側はこのフィールドをゼロに設定し、 BIND の送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドを 1 に設定します。

**downstream_pu_data.pu_sscp_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**downstream_pu_data.pu_sscp_stats.pacing_type**
アップストリーム PU-SSCP セッションで H 用される受信ペーシング・タイプ。このフィールドには、AP_NONE 値を入れることができます。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_DSPU_TEMPLATE

この verb は Communications Server にのみ適用されます。

QUERY_DSPU_TEMPLATE は、暗黙リンクを介して PU 集信にH用される定義済みダウンストリーム PU テンプレートに関する情報を戻します。この情報はリストとして戻されます。特定のダウンストリーム PU テンプレートに関する情報またはいくつかの*固まり* に分けられたリスト情報を入手するには、 **template_name** フィールドを設定する，要があります。そうでなければ ( **list_options** フィールドに AP_FIRST_IN_LIST を設定する場合)、このフィールドは無 k されます。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

## VCB 構造体

```
typedef struct query_dspu_template
{
        unsigned short  opcode;             /* verb operation code            */
        unsigned char   attributes;         /* Verb attributes                */
        unsigned char   reserv2;            /* reserved                       */
        unsigned char   format;             /* format                         */
        unsigned short  primary_rc;         /* primary return code            */
        unsigned long   secondary_rc;       /* secondary return code          */
        unsigned char   *buf_ptr;           /* pointer to buffer              */
        unsigned long   buf_size;           /* buffer size                    */
        unsigned long   total_buf_size;     /* total buffer size required     */
        unsigned short  num_entries;        /* number of entries              */
        unsigned short  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
{
        unsigned short  overlay_size;       /* size of this entry             */
        unsigned char   template_name[8];   /* name of DSPU template          */
        unsigned char   description;        /* resource description           */
        unsigned char   reserv1[12];        /* reserved                       */
        unsigned short  max_instance;       /* max active template instances  */
        unsigned short  active instance;    /* current active instances       */
        unsigned short  num_of_dslu_templates;
                                            /* number of DSLU templates       */
} DSPU_TEMPLATE_DATA;
```

F **dspu_template_data** の後に、 **num_of_dslu_templates** ダウンストリーム LU テンプレートが続きます。 F ダウンストリーム LU テンプレートの形式は以下のとおりです。

```
typedef struct dslu_template_data
{
        unsigned short  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;            /* min NAU address in range       */
```

```
        unsigned char   max_nau;            /* max NAU address in range       */
        unsigned char   reserv1[10];        /* reserved                       */
        unsigned char   host_lu[8];         /* host LU or pool name           */
} DSLU_TEMPLATE;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_DSPU_TEMPLATE

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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** にはヌルを設定する，要があります。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

X定された **template_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から + O されます。

**template_name**
DSPU テンプレートの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**dspu_template_data.overlay_size**
この項目内のバイトの数 (すべてのダウンストリーム LU テンプレートを含む)。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dspu_template_data.template_name**
DSPU テンプレートの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。

**dspu_template_data.description**
q 源の説明 (QUERY_DSPU_TEMPLATE で X 定)。

**dspu_template_data.max_instance**
これは、同 ~ に h 動状態にできるテンプレートのインスタンスの最大数です。

**dspu_template_data.active_instance**
これは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です。

**dspu_template_data.num_of_dslu_templates**
このダウンストリーム PU テンプレートのためのダウンストリーム LU テンプレートの数。このフィールドの後に、フォーカル・ポイント・カテゴリーに登録されている F アプリケーションごとに、 1 つずつ **num_of_dslu_templates_application_id** 項目が入っています。

**dslu_template_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**dslu_template_data.dslu_template.min_nau**
この範囲における NAU アドレスの最小値。

**dslu_template_data.dslu_template.max_nau**
この範囲における NAU アドレスの最大値。

**def_data.allow_timeout**
セッションが、ホスト LU 定義にX定された**タイムアウト**~間中に非アクティブになっている場合に、パーソナル・コミュニケーションズまたは Communications Serverが、このダウンストリーム LU によってH用されるホスト LU をタイムアウトにすることが許可されているかどうかをX定します (AP_YES または AP_NO)。

**def_data.delayed_logon**
ダウンストリーム LU から最初のデータを受信するまで、「プログラム」がダウンストリーム LU とホスト LU との接続を遅らせるかどうかをX定します。その代わり、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 (AP_YES または AP_NO)。

**dslu_template_data.dslu_template.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。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TEMPLATE_NAME

AP_INVALID_LIST_OPTION

関係のある START_NODE パラメーターが設定されていないためにこの verb が実行されない場合は、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ノードが + Oされないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_FOCAL_POINT

QUERY_FOCAL_POINT は、パーソナル・コミュニケーションズまたは Communications Serverが認識しているフォーカル・ポイントに関する情報を戻します。

この情報はリストとして戻されます。特定のフォーカル・ポイント・カテゴリーに関する情報、またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、**ms_category** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

**注:** フォーカル・ポイントが見つからないと、 **fp_data.fp_type** が AP_NO_FP に設定されて 1 つの FP_DATA 構造体が戻されます。以下の構造体を 2 照してください。

## VCB 構造体

```
typedef struct query_focal_point
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   ms_appl_name[8];     /* focal point application name   */
        unsigned char   ms_category[8];      /* focal point category           */
        unsigned char   description[RD_LEN]; /* resource description           */
        unsigned char   fp_fqcp_name[17];    /* focal pt fully qual CP name    */
        unsigned char   bkup_appl_name[8];   /* backup focal pt appl name      */
        unsigned char   bkup_fp_fqcp_name[17];
                                             /* backup FP fully qualified      */
                                             /* CP name                        */
        unsigned char   implicit_appl_name[8];
                                             /* implicit FP 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                       */
        unsigned short  number_of_appls;     /* number of applications         */
} FP_DATA;
```

F **fp_data** の後には、 **number_of_appls** アプリケーション名が続きます。 F アプリケーション名の形式は以下のとおりです。

```
typedef struct application_id
{
        unsigned char    appl_name[8];       /* application name              */
} APPLICATION_ID;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**
AP_QUERY_FOCAL_POINT

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
リスト情報に何を戻すかを示します。つまり、X定された **ms_category** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**ms_category**
管理サービス・カテゴリー。これは、「SNA Management Services」に記述されている管理サービス・カテゴリーの 4 バイトのアーキテクチャー定義値 (右側に EBCDIC スペースが埋め込まれている) の 1 つ、または 8 バイトのタイプ 1134 の EBCDIC インストール先定義名のいずれかです。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**fp_data.overlay_size**
この項目内のバイトの数 (すべてのアプリケーション名を含む)。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**fp_data.ms_appl_name**
現在アクティブになっているフォーカル・ポイント・アプリケーションの名前。これは、「SNA Management Services」に記述されている管理サービス・アプリケーションの 4 バイトのアーキテクチャー定義値 (右側に EBCDIC スペースが埋め込まれている) の 1 つ、または 8 バイトのタイプ 1134 の EBCDIC インストール先定義名のいずれかです。

**fp_data.ms_category**
管理サービス・カテゴリー。これは、「SNA Management Services」に記述されている管理サービス・カテゴリーの 4 バイトのアーキテクチャー定義値 (右側に EBCDIC スペースが埋め込まれている) の 1 つ、または 8 バイトのタイプ 1134 の EBCDIC インストール先定義名のいずれかです。

**fp_data.description**
q 源定義 (DEFINE_FOCAL_POINT で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**fp_data.fp_fqcp_name**
現在アクティブになっているフォーカル・ポイントの完全修飾制御点名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**fp_data.bkup_appl_name**
バックアップ・フォーカル・ポイント・アプリケーションの名前。これは、「SNA Management Services」に記述されている管理サービス・アプリケーションの 4 バイトのアーキテクチャー定義値 (右側に EBCDIC スペースが埋め込まれている) の 1 つ、または 8 バイトのタイプ 1134 の EBCDIC インストール先定義名のいずれかです。

**fp_data.bkup_fp_fqcp_name**
バックアップ・フォーカル・ポイントの完全修飾制御点名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前

は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**fp_data.implicit_appl_name**
暗黙フォーカル・ポイント・アプリケーション名 (DEFINE_FOCAL_POINT verb をH用してX定)。これは、「SNA Management Services」に記述されている管理サービス・アプリケーションの 4 バイトのアーキテクチャー定義値 (右側に EBCDIC スペースが埋め込まれている) の 1 つ、または 8 バイトのタイプ 1134 の EBCDIC インストール先定義名のいずれかです。暗黙フォーカル・ポイントが現行のアクティブ・フォーカル・ポイントであれば、このフィールドは **ms_appl_name** と同じです。

**fp_data.bkup_fp_fqcp_name**
暗黙フォーカル・ポイントの完全修飾制御点名 (DEFINE_FOCAL_POINT verb をH用してX定)。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) 暗黙フォーカル・ポイントが現行のアクティブ・フォーカル・ポイントであれば、このフィールドは **fp_fqcp_name** と同じです。

**fp_data.fp_type**
フォーカル・ポイントのタイプ。詳細については、「*SNA Management Services*」を 2 照してください。値は以下のいずれかになります。

AP_EXPLICIT_PRIMARY_FP
AP_BACKUP_FP
AP_DEFAULT_PRIMARY_FP
AP_IMPLICIT_PRIMARY_FP
AP_DOMAIN_FP
AP_HOST_FP
AP_NO_FP

**fp_data.fp_status**
フォーカル・ポイントの状況。これは、以下のいずれかの値にすることができます。

**AP_NOT_ACTIVE**
フォーカル・ポイントは、現在アクティブではありません。

**AP_ACTIVE**
フォーカル・ポイントは、現在アクティブです。

**AP_PENDING**
フォーカル・ポイントは、保留アクティブです。この状態は、暗黙要求がフォーカル・ポイントに送信されてから応答が受信されるまでの間に発生します。

**AP_NEVER_ACTIVE**
X定されたカテゴリーのアプリケーション登録は受け入れられましたが、そのカテゴリーのフォーカル・ポイント情報がありません。

**fp_data.fp_routing**
MDS 移送をH用してデータをフォーカル・ポイントに送信するときにアプリケーションがX定する経路X定のタイプ。

**AP_DEFAULT**
デフォルト経路X定をH用して MDS_MU をフォーカル・ポイントに送達します。

**AP_DIRECT**
MDS_MU は、セッション中に直接フォーカル・ポイントに経路X定されます。

**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。このフィールドの後には、フォーカル・ポイント・カテゴリーに登録されたFアプリケーションごとに、 **number_of_appls application_id entries** が 1 つずつ続きます。

**appl_name**
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名前。これは、「SNA Management Services」に記述されている管理サービス・アプリケーションの 4 バイトのアーキテクチャー定義値 (右側に EBCDIC スペースが埋め込まれている) の 1 つ、または 8 バイトのタイプ 1134 の EBCDIC インストール先定義名のいずれかです。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_MS_CATEGORY

AP_INVALID_LIST_OPTION

ノードが+ Oされないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_HPR_STATS

この verb は Communications Server にのみ適用されます。

QUERY_HPR_STATS は、ノードの HPR パフォーマンスを記述した統計を戻します。QUERY_HPT_STATS は、 RTP Tower をサポートするノードによってのみサポートされます。

## VCB 構造体

```
typedef struct query_hpr_stats
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned COUNTER num-orig_rs_sent;   /* RS requests sent as origin    */
        unsigned COUNTER num_orig_rs_rej;    /* RS rejections at orign        */
        unsigned COUNTER num_inter_rs_rcvd;  /* Intermediate RS requests      */
        unsigned COUNTER num_inter_rs_rej;   /* Intermediate RS rejections    */
        unsigned COUNTER num_dest_rs_rcvd;   /* RS reqs as destination        */
        unsigned COUNTER num_dest_rs_rej;    /* RS rej sent as destination    */
        unsigned char   reserv[28];          /* reserved                      */
} QUERY_HPR_STATS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_HPR_STATS

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

**num_orig_rs_sent**

このノードで生成された HPR 経路セットアップ要求が、ノード + O後に送信された合計回数。

**num_orig_rs_rej**

このノードで生成された HPR 経路セットアップ要求が、ノード + O後に他のノードによってリジェクトされた合計回数。

**num_inter_rs_rcvd**

HPR 経路セットアップ要求が、ノード + O後に、中間ノードとして動作する
このノードによって処理された合計回数。

**num_inter_rs_rej**

ノード

N 経

# QUERY_ISR_SESSION

この verb は Communications Server にのみ適用されます。

QUERY_ISR_SESSION は、ネットワーク・ノードでのみH用され、ネットワーク・ノードが中間セッション経路X定を提供しているセッションに関するリスト情報を戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のセッションに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **fqpcid** 構造体を設定する , 要があります。そうしないと (**list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、この構造体内のフィールドは無 k されます。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、まず、**fqpcid.pcid** 順に配列され、次に、**fqpcid.fqcp_name** の EBCDIC 辞書配列の順序で行われます。 **fqpcid.pcid_name** 順による配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X定された項目が存在するしないに関係なく)。

**fqpcid** 構造体の形式は、 8 バイトのプロシージャー相関係数 ID (PCID) であり、セッション・オリジネーターのネットワーク修飾 CP 名です。

経路選択制御ベクトル (RSCV) が START_NODE パラメーターにX定されている場合は、 F セッションに関する詳細情報のほかに RSCV も戻されます。この RSVC は、セッションがホップ・バイ・ホップ形式でH用するネットワーク内経路を定義します。

## VCB 構造体

### Format 2

```
typedef struct query_isr_session
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   *buf_ptr;           /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  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;
```

## QUERY_ISR_SESSION

```
typedef struct isr_session_summary
{
        unsigned short  overlay_size;        /* size of this entry            */
        FQPCID          fqpcid;              /* fully qualified procedure     */
                                             /* correlator ID                 */
} ISR_SESSION_SUMMARY;

typedef struct isr_session_detail
{
        unsigned short  overlay_size;        /* size of this entry            */
        FQPCID          fqpcid;              /* fully qualified procedure     */
        unsigned short  sub_overlay_size;    /* offset to appended RSCV       */
                                             /* 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       */
        unsigned char   reserv1[8];          /* reserved                      */
                                             /* resource                      */
        SESSION_STATS   pri_sess_stats;      /* primary hop session stats     */
        SESSION_STATS   sec_sess_stats;      /* secondary hop session         */
                                             /* statistics                    */
        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   */
        unsigned long   rtp_tcid;            /* RTP TC identifier             */
        unsigned long   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 adj CP name     */
        unsigned char   sec_adj_cp_name[17]; /* secondary stage adj CP name   */
        unsigned char   reserv3[3];          /* reserved                      */
        unsigned char   rscv_len;            /* Length of following RSCV      */
} ISR_SESSION_DETAIL;

typedef struct fqpcid
{
        unsigned char   pcid[8];             /* pro correlator identifier     */
        unsigned char   fqcp_name[17];       /* orig's network qualified      */
                                             /* CP name                       */
        unsigned char   reserve3[3];         /* reserved                      */
} FQPCID;

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size       */
        unsigned short  send_ru_size;        /* session send RU size          */
        unsigned short  max_send_btu_size;   /* Maximum send BTU size         */
        unsigned short  max_rcv_btu_size;    /* Maximum rcv BTU size          */
        unsigned short  max_send_pac_win;    /* Max send pacing window size   */
        unsigned short  cur_send_pac_win;    /* Curr send pacing window size  */
        unsigned short  max_rcv_pac_win;     /* Max receive pacing win size   */
        unsigned short  cur_rcv_pac_win;     /* Curr rec pacing window size   */
        unsigned long   send_data_frames;    /* Number of data frames sent    */
        unsigned long   send_fmd_data_frames;
                                             /* num of FMD data frames sent   */
        unsigned long   send_data_bytes;     /* Number of data bytes sent     */
        unsigned long   rcv_data_frames;     /* Num data frames received      */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames recvd  */
        unsigned long   rcv_data_bytes;      /* Num 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[8];         /* Link station name            */
        unsigned char   pacing_type;        /* type of pacing in use        */
} SESSION_STATS;
```

## VCB 構造体

### Format 1 (バック・レベル)

```
typedef struct isr_session_detail
{
        unsigned short  overlay_size;       /* size of this entry           */
        FQPCID          fqpcid;             /* fully qualified procedure    */
        unsigned short  sub_overlay_size;   /* offset to appended RSCV      */
                                            /* 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      */
        unsigned char   reserv1[2];         /* reserved                     */
                                            /* resource                     */
        SESSION_STATS   pri_sess_stats;     /* primary hop session stats    */
        SESSION_STATS   sec_sess_stats;     /* secondary hop session        */
                                            /* statistics                   */
        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  */
        unsigned long   rtp_tcid;           /* RTP TC identifier            */
        unsigned long   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 adj CP name   */
        unsigned char   sec_adj_cp_name[17]; /* secondary stage adj CP name */
        unsigned char   reserv3[3];         /* reserved                     */
        unsigned char   rscv_len;           /* Length of following RSCV     */
} ISR_SESSION_DETAIL;

typedef struct fqpcid
{
        unsigned char   pcid[8];            /* pro correlator identifier    */
        unsigned char   fqcp_name[17];      /* orig's network qualified     */
                                            /* CP name                      */
        unsigned char   reserve3[3];        /* reserved                     */
} FQPCID;

typedef struct session_stats
{
        unsigned short  rcv_ru_size;        /* session receive RU size      */
        unsigned short  send_ru_size;       /* session send RU size         */
        unsigned short  max_send_btu_size;  /* Maximum send BTU size        */
        unsigned short  max_rcv_btu_size;   /* Maximum rcv BTU size         */
        unsigned short  max_send_pac_win;   /* Max send pacing window size  */
        unsigned short  cur_send_pac_win;   /* Curr send pacing window size */
        unsigned short  max_rcv_pac_win;    /* Max receive pacing win size  */
        unsigned short  cur_rcv_pac_win;    /* Curr rec pacing window size  */
        unsigned long   send_data_frames;   /* Number of data frames sent   */
        unsigned long   send_fmd_data_frames;
                                            /* num of FMD data frames sent  */
        unsigned long   send_data_bytes;    /* Number of data bytes sent    */
        unsigned long   rcv_data_frames;    /* Num data frames received     */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames recvd */
        unsigned long   rcv_data_bytes;     /* Num 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[8];         /* Link station name            */
        unsigned char   pacing_type;        /* type of pacing in use        */
} SESSION_STATS;
```

## VCB 構造体

### Format 0 (バック・レベル)

```
typedef struct isr_session_detail
{
   unsigned short  overlay_size;      /* size of this entry           */
   FQPCID          fqpcid;            /* fully qualified procedure    */
   unsigned char   trans_pri;         /* Transmission priority:       */
   unsigned char   cos_name[8];       /* Class-of-service name        */
   unsigned char   ltd_res;           /* Session spans a limited      */
   unsigned char   reserv1[8];        /* reserved                     */
                                      /* resource                     */
   SESSION_STATS   pri_sess_stats;    /* primary hop session stats    */
   SESSION_STATS   sec_sess_stats;    /* secondary hop session        */
                                      /* statistics                   */
   unsigned char   reserv3[3];        /* reserved                     */
   unsigned char   reserva[20];       /* reserved                     */
   unsigned char   rscv_len;          /* Length of following RSCV     */
} ISR_SESSION_DETAIL;
```

**注:** ISR セッション詳細オーバーレーの後に、 *SNA 形式* によって定義された経路選択制御ベクトル (RSCV) が続くことがあります。この制御ベクトルは、ネットワーク内のセッション経路を定義し、BIND で送信されます。この RSCV の組み込みは、ノード + O ~ に決定され (START_NODE のオプションとして)、あとで DEFINE_ISR_STATS をH用して更新することができます。 RSCV を保管するためにこれらの verb をH用しないことをX定する場合は、 **rscv_len** をゼロに設定します。

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_ISR_SESSION

**format**

VCB の形式を識別するほか、戻されたオーバーレーの形式も識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
これは、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

X定された **fqpcid** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**session_type**
この verb は DLUR 維 } セッションを照会しますか、それとも通常の ISR セッションを照会しますか。

AP_ISR_SESSION ISR セッション
AP_DLUR_SESSIONS DLUR セッション

**fqpcid.pcid**
プロシージャー相関係数 ID。これは 8 バイトの 16 進文 z 列です。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**fqpcid.pcid_name**
完全修飾制御点名。この名前は 17 バイトの長さで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**isr_session_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**isr_session_summary.fqpcid.pcid**
プロシージャー相関係数 ID。

**isr_session_summary.fqpcid.fqcp_name**
完全修飾制御点名。この名前は 17 バイトの長さで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**isr_session_detail.overlay_size**
この項目内のバイトの数 (U 加 RSCV を含む)。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**isr_session_detail.sub_overlay_size**
このフィールドは、詳細オーバーレーのサイズを X 定します。RSCV を U 加すると、これが RSCV の + O に対するオフセットになります。このフィールドは、 1 つの詳細構造体の形式のサイズと等しいか、同じにすることができます (将来の H 張が可能です)。

**isr_session_detail.fqpcid.pcid**
プロシージャー相関係数 ID。

**isr_session_detail.fqpcid.fqcp_name**
完全修飾制御点名。この名前は 17 バイトの長さで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**session_detail.trans_pri**
伝送優先順位。以下のいずれかの値に設定されます。

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

**session_detail.cos_name**
サービス・クラス名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。

**session_detail.ltd_res**
セッションが、限定 q 源リンクをH用するかどうかを示します (AP_YES または AP_NO)。

**isr_session_detail.pri_sess_stats.rcv_ru_size**
受信 RU の最大サイズ。

**isr_session_detail.pri_sess_stats.send_ru_size**
送信 RU の最大サイズ。

**isr_session_detail.pri_sess_stats.max_send_btu_size**
1 次セッション・ホップで送信できる最大 BTU サイズ。

**isr_session_detail.pri_sess_stats.max_rcv_btu_size**
1 次セッション・ホップで受信できる最大 BTU サイズ。

**isr_session_detail.pri_sess_stats.max_send_pac_win**
1 次セッション・ホップの送信ペーシング・ウィンドウの最大サイズ。

**isr_session_detail.pri_sess_stats.cur_send_pac_win**
1 次セッション・ホップの送信ペーシング・ウィンドウの現行サイズ。

**isr_session_detail.pri_sess_stats.max_rcv_pac_win**
1 次セッション・ホップの受信ペーシング・ウィンドウの最大サイズ。

**isr_session_detail.pri_sess_stats.cur_rcv_pac_win**
1 次セッション・ホップの受信ペーシング・ウィンドウの現行サイズ。

**isr_session_detail.pri_sess_stats.send_data_frames**
1 次セッション・ホップで送信された通常フロー・データ・フレームの数。

**isr_session_detail.pri_sess_stats.send_data_frames**
1 次セッション・ホップで送信された通常フロー・データ・フレームの数。 DEFINE_ISR_STATS をH用して統計の集合がH用T可にされていると、ゼロがこのフィールドに戻されます。

**isr_session_detail.pri_sess_stats.send_fmd_data_frames**
1 次セッション・ホップで送信された通常フロー FMD データ・フレームの数。 DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.pri_sess_stats.send_data_bytes**
1 次セッション・ホップで送信された通常フロー・データ・バイトの数。 DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.pri_sess_stats.rcv_data_frames**
1 次セッション・ホップで受信された通常フロー・データ・フレームの数。 DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.pri_sess_stats.rcv_fmd_data_frames**
1 次セッション・ホップで受信された通常フロー FMD データ・フレームの数。 DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.pri_sess_stats.rcv_data_bytes**
1 次セッション・ホップで受信された通常フロー・データ・バイトの数。DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.pri_sess_stats.sidh**
セッション ID 上位バイト。

**isr_session_detail.pri_sess_stats.sidl**
セッション ID 下位バイト。

**isr_session_detail.pri_sess_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、BIND の送信側はこのフィールドをゼロに設定します。 BIND 送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドは 1 に設定されます。

**isr_session_detail.pri_sess_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。このフィールドをH用して、セッション統計と、セッション・データが流れるリンクとを関連Uけることができます。

**isr_session_detail.sec_sess_stats.rcv_ru_size**
受信 RU の最大サイズ。

**isr_session_detail.pri_sess_stats.pacing_type**
アップストリーム PU-SSCP セッションでH用される受信ペーシング・タイプ。このフィールドには、 AP_NONE、AP_PACING_FIXED、または AP_PACING_ADAPTIVE の値をX定することができます。

**isr_session_detail.sec_sess_stats.send_ru_size**
送信 RU の最大サイズ。

**isr_session_detail.sec_sess_stats.max_send_btu_size**
2 次セッション・ホップで送信できる最大 BTU サイズ。

**isr_session_detail.sec_sess_stats.max_rcv_btu_size**
2 次セッション・ホップで受信できる最大 BTU サイズ。

**isr_session_detail.sec_sess_stats.max_send_pac_win**
2 次セッション・ホップの送信ペーシング・ウィンドウの最大サイズ。

**isr_session_detail.sec_sess_stats.cur_send_pac_win**
2 次セッション・ホップの送信ペーシング・ウィンドウの現行サイズ。

**isr_session_detail.sec_sess_stats.max_rcv_pac_win**
2 次セッション・ホップ上の受信ペーシング・ウィンドウの最大サイズ。

**isr_session_detail.sec_sess_stats.cur_rcv_pac_win**
2 次セッション・ホップ上の受信ペーシング・ウィンドウの現行サイズ。

**isr_session_detail.sec_sess_stats.send_data_frames**
2 次セッション・ホップ上で送信された通常フロー・データ・フレームの数。DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.sec_sess_stats.send_fmd_data_frames**
2 次セッション・ホップ上で送信された通常フロー FMD データ・フレームの数。DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.sec_sess_stats.send_data_bytes**
2 次セッション・ホップ上で送信された通常フロー・データ・バイトの数。DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.sec_sess_stats.rcv_data_frames**
2 次セッション・ホップ上で受信された通常フロー・データ・フレームの数。DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.sec_sess_stats.rcv_fmd_data_frames**
2 次セッション・ホップ上で受信された通常フロー FMD データ・フレームの数。DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.sec_sess_stats.rcv_data_bytes**
2 次セッション・ホップ上で受信された通常フロー・データ・バイトの数。DEFINE_ISR_STATS をH用して統計の収集をH用T可にすると、ゼロがこのフィールドに戻されます。

**isr_session_detail.sec_sess_stats.sidh**
セッション ID 上位バイト。

**isr_session_detail.sec_sess_stats.sidl**
セッション ID 下位バイト (LFSID からの)。

**isr_session_detail.sec_sess_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、BIND の送信側はこのフィールドをゼロに設定します。 BIND 送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドは 1 に設定されます。

**isr_session_detail.sec_sess_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。このフィールドをH用して、中間セッション統計値と特定のリンク・ステーションとを相関させます。

**isr_session_detail.sec_sess_stats.pacing_type**
1 次セッションでH用される受信ペーシング・タイプ。このフィールドには、AP_NONE、AP_PACING_FIXED、または AP_PACING_ADAPTIVE 値を入れることができます。

**isr_session.detail.sess_lu_type**
BIND にX定されたセッションの LU タイプ。以下のいずれかの値になります。

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
(LU タイプ 5 は、意図的に省略されています。)

DEFINE_ISR_STATS をH用して名前の集合がH用可能にされていない限り、常に AP_LU_TYPE_UNKNOWN が戻されます。

**isr_session.detail.sess_lu_level**

セッションの LU レベル。このフィールドは、以下のいずれかの値になります。

AP_LU_LEVEL_0
AP_LU_LEVEL_1
AP_LU_LEVEL_2
AP_LU_LEVEL_UNKNOWN

6 以 0 の LU タイプの場合、このフィールドは AP_LU_LEVEL_0 に設定されます。 DEFINE_ISR_STATS をH用して名前の集合がH用可能にされていない限り

AP_ISR_INACTIVE
AP_ISR_PENDING_ACTIVE
AP_ISR_ACTIVE
AP_ISR_PENDING_INACTIVE

**isr_session.detail.mode_name**

セッションのモード名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。DEFINE_ISR_STATS をH用して名前の集合がH用可能にされていない限り、常にすべて 2 進ゼロが戻されます。

**isr_session.detail.pri_lu_name**

セッションの 1 次 LU 名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。この名前がH用T可であれば、すべて 2 進ゼロがこのフィールドに戻されます。
DEFINE_ISR_STATS をH用して名前の集合がH用可能にされていない限り、常にすべて 2 進ゼロが戻されます。

**isr_session.detail.sec_lu_name**

セッションの 2 次 LU 名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。この名前がH用T可であれば、すべて 2 進ゼロがこのフィールドに戻されます。
DEFINE_ISR_STATS をH用して名前の集合がH用可能にされていない限り、常にすべて 2 進ゼロが戻されます。

**isr_session.detail.pri_adj_cp_name**

このセッションの 1 次ステージ隣接 CP 名。 1 次セッション・ステージが RTP 接続を通れば、リモート RTP エンドポイントの CP 名が戻されます。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。この名前がH用T可であれば、すべて 2 進ゼロがこのフィールドに戻されます。 DEFINE_ISR_STATS をH用して名前の集合がH用可能にされていない限り、常にすべて 2 進ゼロが戻されます。

々 々 9Z 9Z 々々々々 々 々々々

**isr_session.detail.sec_adj_cp_name**

このセッションの 2 次ステージ隣接 CP 名。 2 次セッション・ステージが RTP 接続を通れば、リモート RTP エンドポイントの CP 名が戻されます。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_FQPCID

AP_INVALID_LIST_OPTION
AP_INVALID_SESSION_TYPE

関係のある START_NODE パラメーターが設定されていないためにこの verb が実行されない場合は、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ノードがネットワーク・ノード・サポートにより構築されていないためにこの verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_INVALID_VERB

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_LOCAL_LU

QUERY_LOCAL_LU は、ローカル LU に関する情報を戻します。
QUERY_LOCAL_LU を発行して、パーソナル・コミュニケーションズまたは Communications Serverの制御点 LU に関する情報を検索することができます。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のローカル LU に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **lu_name** または **lu_alias** フィールドを設定する，要があります。 **lu_name** フィールドが非ゼロであれば、このフィールドは索引の判別にH用されます。 **lu_name** フィールドをすべてゼロに設定すると、 **lu_alias** が索引の判別にH用されます。 **lu_name** と **lu_alias** の両方のフィールドがすべてゼロに設定されると、制御点 (デフォルトの LU)に関連する LU がH用されます。 **list_options** フィールドが AP_FIRST_IN_LIST に設定されると、これらの両方のフィールドは無 k されます。(この場合、 AP_LIST_BY_ALIAS **list_options** が設定されていれば、戻されたリストは LU の別名順に配列されます。それ以 0 の場合は、LU の名前順に配列されます)。リスト形式のH用方法に関する 2 考情報については、10ページの『ノードの照会』 を 2 照してください。

このリストは、X定されたオプションに従って、 **lu_alias** または **lu_name** のいずれかで配列されます。このフィールドは、EBCDIC 辞書配列順に配列されます。

戻されたローカル LU のリストは、関連する PU の名前別にフィルター処理することができます。この場合は、**pu_name** フィールドを設定する，要があります (これ以 0 の場合は、このフィールドをすべてゼロに設定する，要があります)。

## VCB 構造体

### Format 1

```
typedef struct query_local_lu
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   *buf_ptr;           /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;       /* size of this entry           */
        unsigned char   lu_name[8];         /* LU name                      */
        unsigned char   lu_alias[8];        /* LU alias                     */
        unsigned char   description;        /* resource description         */
} LOCAL_LU_SUMMARY;
```

### QUERY_LOCAL_LU

```
typedef struct local_lu_detail
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   lu_name[8];          /* LU name                       */
        LOCAL_LU_DEF_DATA def_data;          /* defined data                  */
        LOCAL_LU_DEF_DATA det_data;          /* determined data               */
} LOCAL_LU_DETAIL;

typedef struct local_lu_def_data
{
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   lu_alias[8];         /* local LU alias                */
        unsigned char   nau_address;         /* NAU address                   */
        unsigned char   syncpt_support;      /* Reserved                      */
        unsigned short  lu_session_limit;    /* LU session limit              */
        unsigned char   default_pool;        /* member of default_lu_pool     */
        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    */
        unsigned char   attach_routing_data[128];
                                             /* routing data for              */
                                             /* incoming attaches             */
        unsigned char   lu_model;            /* LU model name for SDDLU       */
        unsigned char   model_name[8];       /* LU model name for SDDLU       */
        unsigned char   reserv4[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;    /* Is LU-SSCP session active     */
        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
{
        unsigned short  rcv_ru_size;         /* session receive RU size       */
        unsigned short  send_ru_size;        /* session send RU size          */
        unsigned short  max_send_btu_size;   /* max send BTU size             */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size              */
        unsigned short  max_send_pac_win;    /* max send pacing win size      */
        unsigned short  cur_send_pac_win;    /* current send pacing win size  */
        unsigned short  max_rcv_pac_win;     /* max receive pacing win size   */
        unsigned short  cur_rcv_pac_win;     /* current receive pacing        */
                                             /* window size                   */
        unsigned long   send_data_frames;    /* number of data frames sent    */
        unsigned long   send_fmd_data_frames;
                                             /* num of FMD data frames sent   */
        unsigned long   send_data_bytes;     /* number of data bytes sent     */
        unsigned long   rcv_data_frames;     /* num data frames received      */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames recvd  */
        unsigned long   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[8];          /* Link station name             */
        unsigned char   pacing _type;        /* Type of pacing in use         */
} SESSION_STATS;
```

## VCB 構造体

### Format 0

```
typedef struct local_lu_def_data
{
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   lu_alias[8];         /* local LU alias               */
        unsigned char   nau_address;         /* NAU address                  */
        unsigned char   syncpt_support;      /* Reserved                     */
        unsigned short  lu_session_limit;    /* LU session limit             */
        unsigned char   default_pool;        /* member of default_lu_pool    */
        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   */
        unsigned char   attach_routing_data[128];
                                             /* routing data for             */
                                             /* incoming attaches            */
} LOCAL_LU_DEF_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_LOCAL_LU

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

X定された **lu_name** (または、**lu_name** がすべてゼロに設定されている場合は、 **lu_alias**) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**AP_LIST_BY_ALIAS**
戻りリストは、**lu_alias** 順に配列されます。このオプションは、AP_FIRST_IN_LIST がX定されている場合にのみ有効です。AP_LIST_FROM_NEXT または AP_LIST_INCLUSIVE がX定されていれば、リストの配列は、**lu_name** または **lu_alias** のどちらが + O 点として提供されているかによって異なります。

**lu_name**
LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。このフィールドをすべてゼロに設定すると、 **lu_alias** フィールドが索引の判別にH用されます。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**lu_alias**
ローカル定義の LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。 **lu_name** と **lu_alias** の両方のフィールドをすべてゼロに設定すると、制御点 (デフォルトの LU) に関連する LU がH用されます。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**pu_name**
PU 名フィルター。このフィールドをすべてゼロに設定するか、 8 バイト英数 z のタイプ A の EBCDIC 文 z 列に設定し (文 z でO まる)、右側に EBCDIC スペースを埋め込む , 要があります。このフィールドを設定すると、この PU と関連するローカル LU のみが戻されます。すべてゼロに設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**local_lu_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**local_lu_summary.lu_name**
LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。

**local_lu_summary.lu_alias**
ローカル定義の LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**local_lu_summary.description**
q 源の説明 (DEFINE_LOCAL_LU で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**local_lu_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**local_lu_detail.lu_name**
LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。

**local_lu_detail.def_data.description**
q 源の説明 (DEFINE_LOCAL_LU で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**local_lu_detail.def_data.lu_alias**
ローカル定義の LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**local_lu_detail.def_data.nau_address**
LU のネットワーク・アドレス単位アドレス。範囲は 0 〜 255 です。非ゼロ値は、LU が従属 LU であることを暗黙 X 定します。ゼロは、LU が独立 LU であることを暗黙 X 定します。

**local_lu_detail.def_data.syncpt_support**
予約済み。

**local_lu_detail.def_data.lu_session_limit**
ローカル LU のセッションの最大数。ゼロの値は、限度がないことを意味します。

**local_lu_detail.def_data.default_pool**
LU が従属 LU 6.2 デフォルト・プールのメンバーであれば、AP_YES。独立 LU の場合は、常に AP_NO。

**local_lu_detail.def_data.pu_name**
この LU が H 用する PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。このフィールドは、従属 LU によってのみ H 用され、独立 LU の場合はすべて 2 進ゼロに設定されます。

**local_lu_detail.def_data.lu_attributes**
構成済みの LU 属性。このフィールドは、AP_NONE 値を取るか、あるいは互いに OR 結合された以下のオプションの値をとります。

**AP_DISABLE_PWSUB**
ローカル LU がH用T可になっているパスワード置換サポート。

**local_lu_detail.def_data.sscp_id**
このフィールドは、この LU をh動化することが許可された SSCP の ID をX定します。これは 6 バイトの 2 進数フィールドです。このフィールドは従属 LU によってのみH用され、独立 LU の場合、または LU がいずれかの SSCP によってh動化できる場合は、すべて 2 進ゼロに設定しなければなりません。

**local_lu_detail.def_data.disable**
このフィールドは、ローカル LU をH用T可にするか、H用可能にするかを示します。このパラメーターを適切に (AP_YES または AP_NO) 設定して DEFINE_LOCAL_LU を再発行することで、 LU を動的にH用可能にしたりH用T可にしたりできます。H用T可の LU をH用可能にすると、「プログラム」は NOTIFY (online) を発行します。H用可能な LU をH用T可にすると、「プログラム」は NOTIFY (off-line) を発行します。LU がH用T可になっているときにそれをバインドすると、「プログラム」は UNBIND を発行し、続いて NOTIFY (off-line) を発行します。

**local_lu_detail.def_data.attach_routing_data**
このフィールドは、接続の結果、DYNAMIC_LOAD_INDICATION で未変更のまま渡されたデータが、このローカル LU のトランザクション・プログラムに届けられたことを示します。たとえば、このフィールドをH用して、トランザクション・プログラムの作業ディレクトリーへのパスを設定することができます。

**def_data.lu_model**
LU のモデル・タイプと番号。このフィールドは従属 LU によってのみH用され、独立 LU の場合はすべて 2 進ゼロに設定する , 要があります。従属 LU の場合は、以下のいずれかの値に設定されます。

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

従属 LU の場合、**model_name** がすべて 2 進ゼロに設定されていれば、このフィールドは無 k されます。AP_UNKNOWN 以 0 の値がX定され、ホスト・システムが SDDLU (自己定義従属 LU) をサポートしている場合は、ノードは非送信請求 PSID NMVT 応答を生成して、ホストにあるローカル LU を動的に定義します。 PSID サブベクトルには、このフィールドの値に対応するマシン・タイプとモデル番号が入れられます。この verb を再発行することにより、このフィールドを動的に変更することができます。変更結果は、LU がクローズされ非h動化されるまで、有効にはなりません。

**def_data.model_name**
LU のモデル名。このフィールドは従属 LU によってのみH用され、独立 LU の場合はすべて 2 進ゼロに設定する , 要があります。

このフィールドが 2 進ゼロに設定されていて、ホスト・システムが SDDLU をサポートしていれば、ノードは非送信請求 PSID NMVT 応答を生成して、ホストにあるローカル LU を動的に定義します。 PSID サブベクトルには、このフィールドに提供された名前が含まれています。この verb を再発行することにより、このフィールドを動的に変更することができます。変更結果は、LU がクローズされ非 h 動化されるまで、有効にはなりません。

**local_lu_detail.det_data.lu_sscp_session_active**
LU-SSCP セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。 **def_data.nau_address** がゼロの場合は、このフィールドは予約済みになります。

**local_lu_detail.det_data.appl_conn_active**
アプリケーションが LU をH用するかどうかを示します (AP_YES または AP_NO)。**def_data.nau_address** がゼロの場合は、このフィールドは予約済みになります。

**local_lu_detail.det_data.lu_sscp_stats.rcv_ru_size**
このフィールドは常に予約済みです。

**local_lu_detail.det_data.lu_sscp_stats.send_ru_size**
このフィールドは常に予約済みです。

**local_lu_detail.det_data.lu_sscp_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**local_lu_detail.det_data.lu_sscp_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**local_lu_detail.det_data.lu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**local_lu_detail.det_data.lu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**local_lu_detail.det_data.lu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**local_lu_detail.det_data.lu_sscp_stats.cur_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**local_lu_detail.det_data.lu_sscp_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**local_lu_detail.det_data.lu_sscp_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**local_lu_detail.det_data.lu_sscp_stats.send_data_bytes**
送信された通常フロー・データ・バイトの数。

**local_lu_detail.det_data.lu_sscp_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**local_lu_detail.det_data.lu_sscp_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**local_lu_detail.det_data.lu_sscp_stats.rcv_data_bytes**
受信された通常フロー・データ・バイトの数。

**local_lu_detail.det_data.lu_sscp_stats.sidh**
セッション ID 上位バイト。

**local_lu_detail.det_data.lu_sscp_stats.sidl**
セッション ID 下位バイト。

**local_lu_detail.det_data.lu_sscp_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、このフィールドをゼロに設定し、ACTLU 送信側に 2 次リンク・ステーションが含まれていれば、このフィールドを 1 に設定します。

**local_lu_detail.det_data.lu_sscp_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。このフィールドを H 用すれば、このセッションと、このセッションが通るリンクとを相関させることができます。

**注:** LU-SSCP 統計 (**local_lu_detail.det_data.lu_sscp_stats**) は、**nau_address** がゼロでない場合にのみ有効です。これ以 0 の場合は、これらのフィールドは予約済みです。

**local_lu_detail.det_data.lu_sscp_stats.pacing_type**
LU-SSCP セッションで H 用される受信ペーシング・タイプ。これは AP_NONE に設定されます。

**local_lu_detail.det_data.sscp_id**
これは 6 バイトのフィールドで、このフィールドには、この LU で H 用された PU の ACTPU で受信された SSCP ID が含まれています。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_ALIAS

AP_INVALID_LU_NAME
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_LOCAL_TOPOLOGY

すべての APPN ノードは、すべての隣接ノードへの伝送グループ (TG) に関する情報を } つローカル・トポロジー・データベースを維 } します。

QUERY_LOCAL_TOPOLOGY をH用すれば、これらの TG に関する情報を戻すことができます。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のローカル TG に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **dest**、**dest_type**、および **tg_num** フィールドを設定する , 要があります。これ以 0 の場合 (**list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合) は、これらのフィールドは無 k されます。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。このリストは、まず **dest** で配列され、次に、**dest_type** で配列され、最後に **tg_num** で配列されます。 **dest** 名は、まず、名前の長さ順に配列され、名前の長さが同じ場合は、次に、辞書配列の順序で行われます。**dest_type** フィールドは、AP_LEN_NODE、AP_NETWORK_NODE、AP_END_NODE, AP_VRN の順序に従います。 **tg_num** は数 z 順に配列されます。

AP_LIST_INCLUSIVE を選択すると、戻りリストはその名前をもつ最初の有効なレコードから + O されます。

AP_LIST_FROM_NEXT を選択すると、リストは、 X定された名前の次の名前をもつ最初の有効なレコードから + O されます。

## VCB 構造体

```
typedef struct query_local_topology
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   *buf_ptr;           /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;       /* size of this 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
{
        unsigned short  overlay_size;       /* size of this 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            */
        unsigned long   rsn;                /* resource sequence number       */

        unsigned char   status;             /* TG status                      */
        TG_DEFINED_CHARS tg_chars;          /* TG characteristics             */
        unsigned char   cp_cp_session_active;
                                            /* CP-CP session is active        */
        unsigned char   branch_tg;          /* branch link type               */
        unsigned char   reserva[13];        /* reserved                       */
} LOCAL_TOPOLOGY_DETAIL;

typedef struct link_address
{
        unsigned short  length;             /* length                         */
        unsigned short  reserve1;           /* reserved                       */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                            /* address                        */
} LINK_ADDRESS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_LOCAL_TOPOLOGY

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

**dest**、**dest_type**、および **tg_num** を組み合わせたX定 (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**dest** TG の完全修飾宛先ノード名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**dest_type**
この TG の宛先ノードのノード・タイプ。これは、以下のいずれかの値にすることができます。

AP_NETWORK_NODE
AP_VRN
AP_END_NODE
AP_LEARN_NODE

**dest_type** がT明の場合は、 AP_LEARN_NODE をX定する , 要があります。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**tg_num**
TG と関連する番号。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**local_topology_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**local_topology_summary.dest**
TG の完全修飾宛先ノード名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**local_topology_summary.dest_type**
この TG の宛先ノードのタイプ。以下のいずれかの値に設定されます。

AP_NETWORK_NODE
AP_VRN
AP_END_NODE

**dest_type** を AP_END_NODE に設定すると、 TG 宛先が LEN ノードまたはエンド・ノードのいずれかに X 定されることになる点に注意してください。

**local_topology_summary.tg_num**
TG と関連する番号。

**local_topology_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**local_topology_detail.dest**
TG の完全修飾宛先ノード名。この名前の長さは 17 バイトで、右側に EBCDIC スペースが埋め込まれています。この名前は、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなっています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**local_topology_detail.dest_type**
この TG の宛先ノードのタイプ。以下のいずれかの値に設定されます。

AP_NETWORK_NODE
AP_VRN
AP_END_NODE

**dest_type** を AP_END_NODE に設定すると、 TG 宛先が LEN ノードまたはエンド・ノードのいずれかに X 定されることになる点に注意してください。

**local_topology_detail.tg_num**
TG と関連する番号。

**local_topology_detail.dlc_data.length**
VRN への接続の DLC アドレスの長さ (**dest_type** が AP_VRN でなければ、ゼロに設定されます)。

**local_topology_detail.dlc_data.address**
VRN への接続の DLC アドレス。

**local_topology_detail.rsn**
q 源シーケンス番号。これは、この q 源を所有するネットワーク・ノードによってd り当てられます。

**local_topology_detail.status**
TG の状況をX定します。これは、以下のいずれかの値、または複数の値を OR 結合した値になります。

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_TG_HPR
AP_TG_RTP
AP_NONE

**local_topology_detail.tg_chars**
TG 特性 (33ページの『DEFINE_CN』 を 2 照)。

**local_topology_detail.cp_cp_session_active**
ローカル・ノードの競合勝者 CP-CP セッションがアクティブになっているかどうかをX定します (AP_NO または AP_YES)。

**local_topology_detail.branch_link_type**
BrNN のみ。この TG のこのブランチ・リンク・タイプ。これは、以下のいずれかに設定されます。

**AP_UPLINK**
このリンクはアップリンクです。

**AP_DOWNLINK**
このリンクは、EN に対してダウンリンクです。

**AP_DOWNLINK_TO_BRNN**
この TG は、EN フェースを示す BrNN に対してダウンリンクです。

**AP_OTHERLINK**
このリンクはアザーリンクです。

その他のノード・タイプ: このフィールドは意味がなく、常に AP_BRNN_NOT_SUPPORTED に設定されています。

**local_topology_detail.branch_tg**
NN のみ。この TG がブランチ TG であるかどうかをX定します。

**AP_NO**
この TG はブランチ TG ではありません。

**AP_YES**
この TG はブランチ TG です。

その他のノード・タイプ: このフィールドは意味がなく、常に AP_NO に設定されています。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TG

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_LS

QUERY_LS は、ノードで定義されたリンク・ステーションに関する情報のリストを戻します。この情報は、『決定済みデータ』 (実行中に動的に収集されたデータ) および『定義済みデータ』 (DEFINE_LS のアプリケーションによって提供されたデータ) として構造化されます。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の LS に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **ls_name** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**ls_name** 順に配列されます。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。 AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

戻されたリンク・ステーションのリストは、自分が属するポートの名前別にフィルター処理することができます。この場合は、**port_name** フィールドを設定する , 要があります (それ以 0 の場合は、このフィールドをすべてゼロに設定する , 要があります)。

## VCB 構造体

### Format 1

```
typedef struct query_ls
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* Verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* Primary return code           */
        unsigned long   secondary_rc;        /* Secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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];        /* name of link station          */
} QUERY_LS;

typedef struct ls_summary
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   ls_name[8];          /* link station name             */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   dlc_type;            /* DLC type                      */
        unsigned char   state;               /* link station state            */
        unsigned short  act_sess_count;      /* currently active sess count   */
        unsigned char   det_adj_cp_name[17]; /* determined adj CP name        */
        unsigned char   det_adj_cp_type;     /* determined adj 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
{
        unsigned short  overlay_size;        /* size of this entry           */
        unsigned char   ls_name[8];          /* link stations name           */
        LS_DET_DATA     det_data;            /* determined data              */
        LS_DEF_DATA     def_data;            /* defined data                 */
} LS_DETAIL;

typedef struct ls_det_data
{
        unsigned short  act_sess_count;      /* curr 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;             /* is LS is dynamic ?           */
        unsigned char   migration;           /* supports migration partners  */
        unsigned char   tg_num;              /* TG number                    */
        LS_STATS        ls_stats;            /* link station statistics      */
        unsigned long   start_time;          /* time LS started              */
        unsigned long   stop_time;           /* time LS stopped              */
        unsigned long   up_time;             /* total time LS active         */
        unsigned long   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 activate                */
        unsigned char   ls_role;             /* link station role            */
        unsigned char   reserva;             /* reserved                     */
        unsigned char   node_id[4];          /* determined node id           */
        unsigned short  active_isr_count;    /* currently active ISR sessions */
        unsigned short  active_lu_sess_count;
                                             /* active LU-LU session count   */
        unsigned short  active_sscp_sess_count;
                                             /* active SSCP session count    */
        ANR_LABEL       reverse_anr_labe;    /* reverse ANR label            */
        unsigned short  max_send_btu_size;   /* negotiated max BTU length    */
        unsigned char   brnn_link_type;      /* branch link type             */
        unsigned char   adj_cp_is_brnn;      /* adjacent CP is a BrNN        */
        unsigned char   reservb[6];          /* reserved                     */
} LS_DET_DATA;

typedef struct anr_label
{
        unsigned short  length;              /* ANR label length             */
        unsigned short  reserv;              /* reserved                     */
        unsigned char   label[MAX_ANR_LABEL_SIZE];
                                             /* ANR label                    */
} ANR_LABEL;

typedef struct ls_def_data
{
        unsigned char   description[RD_LEN]; /* resource description         */
        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 flag */
        unsigned char   dspu_services;  /* Services provided for          */
                                        /* downstream PU                  */
        unsigned char   dspu_name[8];   /* Downstream PU name (reserved   */
                                        /* if dspu_services is set to     */
                                        /* AP_NONE or AP_DLUR)            */
        unsigned char   dlus_name[17];  /* DLUS name if dspu_services     */
                                        /* is set to AP_DLUR              */
        unsigned char   bkup_dlus_name[17]; /* Backup DLUS name if        */
                                        /* dspu_services is set           */
                                        /* to AP_DLUR                     */
        unsigned char   hpr_supported;  /* does the link support HPR?     */
        unsigned char   hpr_link_lvl_error; /* does the link support HPR  */
                                        /* link-level error recovery?     */
        unsigned short  link_deact_timer;   /* HPR link deactivation timer */
        unsigned char   reserv1;        /* reserved                       */
        unsigned char   default_nn_server;  /* Use as 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             */
        unsigned short  target_pacing_count; /* target pacing count       */
        unsigned short  max_send_btu_size;  /* max send BTU size          */
        unsigned char   ls_role;        /* link station role to use       */
                                        /* on this link                   */
        unsigned char   max_ifrm_rcvd;  /* max number of I-frames rcvd    */
        unsigned short  dlus_retry_timeout; /* DLUS retry timeout         */
        unsigned short  dlus_retry_limit;   /* DLUS retry limit           */
        unsigned char   conventional_lu_compression;
                                        /* Data compression requested for */
                                        /* conventional LU sessions       */
        unsigned char   conventional_lu_cryptography;
                                        /* Cryptography required for      */
                                        /* conventional LU sessions       */
        unsigned char   reserv3;        /* reserved                       */
        unsigned char   retry_flags;    /* conditions for automatic       */
                                        /* retries                        */
        unsigned short  max_activation_attempts;
                                        /* how many automatic retries:    */
        unsigned short  activation_delay_timer;
                                        /* delay between automatic        */
                                        /* retries                        */
        unsigned char   branch_link_type;   /* branch link type           */
        unsigned char   adj_brnn_cp_support; /* adjacent BrNN CP support  */
        unsigned char   reserv4[20];    /* reserved                       */
        unsigned short  link_spec_data_len; /* length of link specific data */
} LS_DEF_DATA;

typedef struct link_address
{
        unsigned short  length;         /* length                         */
        unsigned short  reserve1;       /* reserved                       */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                        /* address                        */
} LINK_ADDRESS;

typedef struct link_spec_data
{
            unsigned char  link_data[SIZEOF_LINK_SPEC_DATA];

}  LINK_SPEC_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;         /* Modem class                   */
        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
{
        unsigned long   in_xid_bytes;        /* number of XID bytes received  */
        unsigned long   in_msg_bytes;        /* num message bytes received    */
        unsigned long   in_xid_frames;       /* num XID frames received       */
        unsigned long   in_msg_frames;       /* num message frames received   */
        unsigned long   out_xid_bytes;       /* num XID bytes sent            */
        unsigned long   out_msg_bytes;       /* num message bytes sent        */
        unsigned long   out_xid_frames;      /* num XID frames sent           */
        unsigned long   out_msg_frames;      /* num message frames sent       */
        unsigned long   in_invalid_sna_frames;
                                             /* num invalid frames received   */
        unsigned long   in_session_control_frames;
                                             /* num control frames received   */
        unsigned long   out_session_control_frames;
                                             /* num control frames sent       */
        unsigned long   echo_rsps;           /* response from adj LS count    */
        unsigned long   current_delay;       /* time taken for last test sig  */
        unsigned long   max_delay;           /* max delay by test signal      */
        unsigned long   min_delay;           /* min delay by test signal      */
        unsigned long   max_delay_time;      /* time since longest delay      */
        unsigned long   good_xids;           /* successful XID on LS count    */
        unsigned long   bad_xids;            /* unsuccessful XID on LS count  */
} LS_STATS;
```

## VCB 構造体

### Format 0 (バック・レベル)

```
typedef struct ls_det_data
{
        unsigned short  act_sess_count;      /* curr 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;             /* is LS is dynamic ?            */
        unsigned char   migration;           /* supports migration partners   */
        unsigned char   tg_num;              /* TG number                     */
        LS_STATS        ls_stats;            /* link station statistics       */
        unsigned long   start_time;          /* time LS started               */
        unsigned long   stop_time;           /* time LS stopped               */
        unsigned long   up_time;             /* total time LS active          */
        unsigned long   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 activate                 */
```

```
        unsigned char   ls_role;            /* link station role             */
        unsigned char   reserva;            /* reserved                     */
        unsigned char   node_id[4];         /* determined node id           */
        unsigned short  active_isr_count;   /* currently active ISR sessions  */
        unsigned char   reservb[30];        /* reserved                     */
} LS_DET_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_LS

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB の Format 1 バージョンをX定するには、このフィールドを 1 に設定します。これを 0 に設定すると、「プログラム」は Format 0 の LS_DET_DATA 構造体を戻します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

X定された **ls_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + Oされます。

**ls_name**
リンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**port_name**
ポート名フィルター。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。このフィールドを設定すると、このポートに属するリンク・ステーションのみが戻されます。すべてゼロに設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**ls_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**ls_summary.ls_name**
リンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**ls_summary.description**
q 源の説明 (DEFINE_LS でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**ls_summary.dlc_type**
DLC のタイプ。パーソナル・コミュニケーションズまたは Communications Serverは、以下のタイプをサポートします。

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

DEFINE_DLC verb に新しいタイプをX定することによって、追加の DLC タイプを定義することができます。詳細については、 49ページの『DEFINE_DLC』 を2照してください。

**ls_summary.state**
このリンク・ステーションの状態。このフィールドは以下のいずれかの値に設定されます。

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

**ls_summary.act_sess_count**
リンクをH用するアクティブ・セッションの合計数 (エンドポイントと中間の両方)。

**ls_summary.det_adj_cp_name**
リンクh動化中に判別された完全修飾の 17 バイトの隣接 CP 名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文z列からなり、右側に EBCDIC スペースが埋め込まれています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) LS が非アクティブの場合は、ヌルになります。

**ls_summary.adj_cp_type** が AP_NETWORK_NODE、AP_END_NODE、AP_APPN_NODE、または AP_BACK_LEVEL_LEN_NODE のいずれでもない場合、このフィールドは予約済みになります。

**ls_summary.det_adj_cp_type**
リンクh動化中に判別された隣接ノードのタイプ。以下のいずれかの値になります。

AP_END_NODE
AP_NETWORK_NODE
AP_LEARN_NODE
AP_VRN

LS が非アクティブであれば、これは AP_LEARN_NODE になります。

**ls_summary.adj_cp_type** が AP_NETWORK_NODE、AP_END_NODE、AP_APPN_NODE、または AP_BACK_LEVEL_LEN_NODE のいずれでもない場合、このフィールドは予約済みになります。

**ls_summary.port_name**
このリンク・ステーションと関連するポートの名前。これは、ローカル=示可能文zセットの 8 バイトの文z列です。8 バイトすべてが有効です。

**ls_summary.adj_cp_name**

17 バイトの完全修飾隣接制御点名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) これは、暗黙リンクの場合、ヌルになります。

**ls_summary.adj_cp_type**

隣接ノードのタイプ。以下のいずれかの値になります。

AP_END_NODE
AP_NETWORK_NODE
AP_APPN_NODE
AP_BACK_LEVEL_LEN__NODE
AP_HOST_XID3
AP_HOST_XID0
AP_DSPU_XID
AP_DSPU_NOXID

**ls_detail.overlay_size**

この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**ls_detail.ls_name**

リンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**ls_detail.det_data.act_sess_count**

リンクを H 用するアクティブ・セッションの合計数 (エンドポイントと中間の両方)。

**ls_detail.det_data.dlc_type**

DLC のタイプ。パーソナル・コミュニケーションズまたは Communications Serverは、以下のタイプをサポートします。

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

DEFINE_DLC verb に新しいタイプを X 定することによって、追加の DLC タイプを定義することができます。詳細については、 49 ページの『DEFINE_DLC』 を 2 照してください。

**ls_detail.det_data.state**

このリンク・ステーションの状態。このフィールドは以下のいずれかの値に設定されます。

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

**ls_detail.det_data.sub_state**

このフィールドは、このリンク・ステーションの状態に関するより詳細な情報を提供します。このフィールドは以下のいずれかの値に設定されます。

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**

17 バイトの完全修飾隣接制御点名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**ls_summary.adj_cp_type** が AP_NETWORK_NODE、AP_END_NODE、AP_APPN_NODE、または AP_BACK_LEVEL_LEN_NODE のいずれでもない場合、このフィールドは予約済みになります。

**ls_detail.det_data.det_adj_cp_type**

リンク h 動化中に判別された隣接ノードのタイプ。以下のいずれかの値になります。

AP_END_NODE
AP_NETWORK_NODE
AP_LEARN_NODE
AP_VRN

**ls_summary.adj_cp_type** が AP_NETWORK_NODE、AP_END_NODE、AP_APPN_NODE、または AP_BACK_LEVEL_LEN_NODE のいずれでもない場合、このフィールドは予約済みになります。

**ls_detail.det_data.dlc_name**

DLC の名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**ls_detail.det_data.dynamic**

リンクが明示的に定義されたか (DEFINE_LS コマンドによって)、あるいは暗黙的または動的に定義されたか (隣接ノードからの接続要求に応答するか、または接続ネットワークを介して別のノードに動的に接続することによって) をX定します。これは、AP_YES または AP_NO になります。

**ls_detail.det_data.migration**

隣接ノードが移行レベル・ノードであるか (たとえば、ローエントリー・ネットワーキング (LEN) ノード)、フル APPN ネットワーク・ノードであるか、エンド・ノードであるか (AP_YES、AP_NO、または AP_UNKNOWN) をX定します。

**ls_detail.det_data.tg_num**
TG と関連する番号。

**ls_detail.det_data.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_detail.det_data.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_detail.det_data.ls_stats.in_xid_frames**
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_detail.det_data.ls_stats.in_msg_frames**
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_detail.det_data.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_detail.det_data.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_detail.det_data.ls_stats.out_xid_frames**
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_detail.det_data.ls_stats.out_msg_frames**
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_detail.det_data.ls_stats.in_invalid_sna_frames**
このリンク・ステーションで受信された SNA 無効フレームの合計数。

**ls_detail.det_data.ls_stats.in_session_control_frames**
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_detail.det_data.ls_stats.out_session_control_frames**
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_detail.det_data.ls_stats.echo_rsps**
隣接ノードから受信されたエコー応答の数。エコー要求は、伝搬遅延を測定するために定期的に隣接ノードに送信されます。

**ls_detail.det_data.ls_stats.current_delay**
最後のテスト信号がこのリンク・ステーションから隣接リンク・ステーションへ送信されてから戻ってくるまでにかかった～間 (ミリ C)。

**ls_detail.det_data.ls_stats.max_delay**
テスト信号がこのリンク・ステーションから隣接リンク・ステーションへ送信されてから戻ってくるまでにかかった最長～間 (ミリ C 単位)。

**ls_detail.det_data.ls_stats.min_delay**
テスト信号がこのリンク・ステーションから隣接リンク・ステーションへ送信されてから戻ってくるまでにかかった最短～間 (ミリ C 単位)。

**ls_detail.det_data.ls_stats.max_delay_time**
システム + O 以降に最長遅延が発生した～刻 (1/100 C 単位)。

**ls_detail.det_data.ls_stats.good_xids**
このリンク・ステーションの + O 以降、そこで行われた成功 XID 交換の合計数。

**ls_detail.det_data.ls_stats.bad_xids**
このリンク・ステーションの + O 以降、そこで行われた失敗 XID 交換の合計数。

**ls_detail.det_data.start_time**
システム + O 以降、リンク・ステーションが最後に h 動化された (つまり、モード設定コマンドが完了した) ~ 刻 (1/100 C 単位)。

**ls_detail.det_data.stop_time**
システム + O 以降、リンク・ステーションが最後に非 h 動化された ~ 刻 (1/100 C 単位)。

**ls_detail.det_data.up_time**
システム + O 以降、このリンク・ステーションがアクティブ状態になっていた合計 ~ 間 (1/100 C 単位)。

**ls_detail.det_data.current_state_time**
このリンク・ステーションが現在の状態になっていた合計 ~ 間 (1/100 C 単位)。

**ls_detail.det_data.deact_cause**
リンク・ステーションが最後に非 h 動化された原因。このフィールドは、以下のいずれかの値に設定されます。

**AP_NONE**
リンク・ステーションが非 h 動化されたことはありません。

**AP_DEACT_OPER_ORDERLY**
オペレーターからの順序 STOP コマンドにより、リンク・ステーションが非 h 動化されました。

**AP_DEACT_OPER_IMMEDIATE**
オペレーターからの即 ~ STOP コマンドにより、リンク・ステーションが非 h 動化されました。

**AP_DEACT_AUTOMATIC**
リンク・ステーションが自動的に非 h 動化されました (たとえば、リンク・ステーションを H 用するセッションがなくなったため)。

**AP_DEACT_FAILURE**
障 2 のためにリンク・ステーションが非 h 動化されました。

**ls_detail.det_data.hpr_support**
ローカル・ノードと隣接ノードの能力を考慮に入れて、 TG でサポートされる高性能経路 X 定 (HPR) のレベル (つまり、AP_NONE、AP_BASE、または AP_RTP)。

**ls_detail.det_data.anr_label**
ローカル・リンクに d り振られた HPR 自動ネットワーク経路 X 定 (ANR) ラベル。

**ls_detail.det_data.hpr_link_lvl_error**
リンク上の HPR トラフィックのためにリンク・レベル・エラー回 | を H 用するかどうかを示します。

**ls_detail.def_data.auto_act**
現在リンクで、リモート h 動化とオンデマンド h 動化のうち、どちらがH用可能になっているかをX定します。以下の値が戻されます (OR で結合することができます)。

**AP_AUTO_ACT**
リンクは、ローカル・ノードによるオンデマンドの h 動化が可能です。

**AP_REMOTE_ACT**
リンクは、リモート・ノードによる h 動化が可能です。

**ls_detail.det_data.ls_role**
このリンク・ステーションでのリンク・ステーション・ロール。これは、最初、このリンク・ステーションに対して定義されたリンク・ステーション・ロールに設定されます。定義されたロールが折衝可能であれば、この値は、XID 交換中に折衝済みのロール (1 次または 2 次) に変更され、リンクが非 h 動化されたときに、折衝可能なロールに戻されます。

**AP_LS_NEG**
リンク・ステーション・ロールが折衝可能です。

**AP_LS_PRI**
リンク・ステーション・ロールが 1 次です。

**AP_LS_SEC**
リンク・ステーション・ロールが 2 次です。

**ls_detail.det_data.node_id**
XID 交換中に隣接ノードから受信されたノード ID。これは、4 バイトの16 進数文 z 列です。

**ls_detail.det_data.active_isr_count**
リンクをH用するアクティブ中間セッションの数。

**ls_detail.det_data.active_lu_sess_count**
リンクをH用するアクティブ LU-LU セッションの数。

**ls_detail.det_data.active_sscp_sess_count**
リンクをH用するアクティブ LU-SSCP および PU-SSCP セッションの数。

**ls_detail.det_data.reverse_anr_label.length**
リンク・ステーションの逆自動ネットワーク経路X定 (anr) ラベルの長さ。リンクが HPR をサポートしていない場合、またはラベルが認識できない場合は、このフィールドはゼロになります。

**ls_detail.det_data.local_address**
このリンク・ステーションのローカル・アドレス。

**ls_detail.det_data.max_send_btu_size**
このリンクで送信できる最大 BTU サイズ (隣接ノードとの折衝で決定されます)。リンクの h 動化がまだ行われていない場合は、ゼロが戻されます。

**ls_detail.det_data.brnn_link_type**
BrNN のみ。このブランチのリンク・タイプ。以下のいずれかの値になります。

**AP_UPLINK**
このリンクはアップリンクです。

**AP_DOWNLINK**
このリンクはダウンリンクです。

**AP_OTHERLINK**
このリンクはアザーリンクです。

**AP_UNKNOWN_LINK_TYPE**
このリンクはアザーリンクです。

**AP_BRNN_NOT_SUPPORTED**
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。

その他のノード・タイプ: このフィールドは意味がなく、常に AP_BRNN_NOT_SUPPORTED に設定されています。

**ls_detail.det_data.adj_cp_is_brnn**
すべてのノード・タイプ: 隣接ノードが BrNN であるかどうかを X 定します。

**AP_UNKNOWN**
隣接ノードが BrNN であるかどうかは T 明です。

**AP_NO**
隣接ノードは BrNN ではありません。

**AP_YES**
隣接ノードは BrNN です。

**ls_detail.def_data.description**
q 源の説明 (DEFINE_LS で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**ls_detail.def_data.port_name**
このリンク・ステーションと関連するポートの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。リンクが VRN に対するものであれば、このフィールドは、 VRN との接続に H 用する実ポートの名前 (DEFINE_CN verb で X 定) を X 定します。

**ls_detail.def_data.adj_cp_name**
17 バイトの完全修飾隣接制御点名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) この名前は、**back_lvl_len_end_node** が AP_NO に設定されていない場合、またはリンク・ステーションと関連するポートの切り替えが定義されている場合に定義されます。

**ls_detail.def_data.adj_cp_type**
隣接ノード・タイプ。

**AP_NETWORK_NODE**
ノードが APPN ネットワーク・ノードであることを X 定します。

**AP_END_NODE**
ノードが APPN エンド・ノードまたは上位レベル LEN ノードであることを X 定します。

**AP_APPN_NODE**
ノードが APPN ネットワーク・ノード、APPN エンド・ノード、または上位レベル LEN ノードであることをX定します。ノード・タイプは、XID 交換~にN認されます。

**AP_BACK_LEVEL_LEN_NODE**
ノードがバック・レベル LEN ノードであることをX定します。

**AP_HOST_XID3**
ノードがホストであり、ノード・オペレーター機能が 3 XID 形式のノードからのポーリング XID に応答することをX定します。

**AP_HOST_XID0**
ノードがホストであり、ノード・オペレーター機能が 0 XID 形式のノードからのポーリング XID に応答することをX定します。

**AP_DSPU_XID**
ノードがダウンストリーム PU であり、ノード・オペレーター機能にリンクh動化~の XID 交換が含まれることをX定します。

**AP_DSPU_NOXID**
ノードがダウンストリーム PU であり、ノード・オペレーター機能にリンクh動化~の XID 交換が含まれないことをX定します。

**注:** VRN へのリンク・ステーションは、常に動的であるため定義されません。

**ls_detail.def_data.dest_address.length**
隣接ノードでの宛先リンク・ステーションのアドレスの長さ。

**ls_detail.def_data.dest_address.address**
隣接ノードでのリンク・ステーションの宛先アドレス。

**ls_detail.def_data.auto_act_supp**
リンクが、START_LS verb によって+Oされた後に自動的にh動化され、STOP_LS によって停_されるかどうかを示します (AP_YES または AP_NO)。

**ls_detail.def_data.tg_number**
前もってd り当てられた TG 番号 (1 から 20 までの範囲)。この番号をH用して、リンクがh動化されたときのリンクを示します。ゼロは、 TG 番号がv前d り当てされておらず、リンクh動化~に折衝されることを示します。

**ls_detail.def_data.limited_resource**
リンクをH用するセッションがない場合に、このリンク・ステーションを非h動化するかどうかをX定します。以下のいずれかの値に設定されます。

**AP_NO**
このリンクは限定されたq源ではなく、自動的に非h動化されることはありません。

**AP_YES or AP_NO_SESSIONS**
このリンクは限定されたq源であり、このリンクをH用するアクティブ・セッションがない場合に自動的に非h動化されます。

**AP_INACTIVITY**
このリンクは限定されたq源であり、このリンクをH用するアクテ

ィブ・セッションがない場合、もしくは **link_deact_timer** フィールドでX定した期間、リンク上でデータのフローがない場合に、自動的に非h動化されます。

**ls_detail.def_data.solicit_sscp_sessions**
AP_YES は、SSCP と、ローカル制御点および従属 LU との間のセッションを+Oするようホストに要求します。 AP_NO は、このリンクでの SSCP とのセッションを要求しません。

**ls_detail.def_data.pu_name**
**solicit_sscp_sessions** が AP_YES に設定されている場合にこのリンクをH用するローカル PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文z列 (文zでOまる) で、右側に EBCDIC スペースが埋め込まれています。**solicit_sscp_sessions** が AP_NO に設定されていれば、このフィールドは予約済みになります。

**ls_detail.def_data.disable_remote.act**
このリンクのリモートh動化がサポートされているかどうかをX定します (AP_YES または AP_NO)。

**ls_detail.def_data.dspu_services**
**solicit_sscp_sessions** が AP_NO に設定されている場合に、ローカル・ノードがこのリンクを介してダウンストリーム PU に提供するサービスをX定します。これは、以下のいずれかの値に設定されます。

**AP_PU_CONCENTRATION**
ローカル・ノードは、ダウンストリーム PU に PU 集信を提供します。

**AP_DLUR**
ローカル・ノードは、ダウンストリーム PU に DLUR サービスを提供します。

**AP_NONE**
ローカル・ノードは、このダウンストリーム PU のためのサービスを提供しません。

**solicit_sscp_sessions** が AP_YES に設定されていれば、このフィールドは予約済みになります。

**ls_detail.def_data.dspu_name**
ダウンストリーム PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文z列 (文zでOまる) で、右側に EBCDIC スペースが埋め込まれています。これは、**solicit_sscp_sessions** が AP_NO に設定される場合にのみ有効です。

**ls_detail.def_data.dlus_name**
ダウンストリーム・ノードへのリンクがh動化されるときから、 DLUR が SSCP サービスを送信請求する DLUS ノードの名前。この値は、すべてゼロに設定されるか、または 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文z列からなり、右側に EBCDIC スペースが埋め込まれている 17 バイトの文z列に設定されます。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) このフィールドがすべてゼロに設定されていれば、リンクがh動化されるときに、デフォルトのグローバル

DLUS (DEFINE_DLUR_DEFAULTS verb で定義されている場合) が送信請求されます。**dlus_name** がゼロに設定され、かつデフォルトのグローバル DLUS がなければ、リンクが h 動化されるときに DLUR は SSCP の接続を + O しません。 **dspu_services** が AP_DLUR に設定されていなければ、このフィールドは予約済みになります。

**ls_detail.def_data.bkup_dlus_name**
ダウンストリーム PU のバックアップとして H 用される DLUS ノードの名前。この値は、すべてゼロに設定されるか、または 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれている 17 バイトの文 z 列に設定されます。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) このフィールドがすべてゼロに設定されていれば、デフォルトのグローバル・バックアップ DLUS (DEFINE_DLUR_DEFAULTS verb で定義されている場合) がこの PU のバックアップとして H 用されます。 **dspu_services** が AP_DLUR に設定されていなければ、このフィールドは予約済みになります。

**ls_detail.def_data.hpr_supported**
HPR がこのリンク上でサポートされているかどうかを X 定します (AP_YES または AP_NO)。

**ls_detail.def_data.hpr_link_lvl_error**
HPR リンク・レベル・エラー回 | タワーがこのリンクでサポートされているかどうかを X 定します (AP_YES または AP_NO)。 **hpr_supported** が AP_NO に設定されていれば、このパラメーターが予約済みになることに注意してください。

**ls_detail.def_data.link_deact_timer**
限定 q 源リンク非 h 動化タイマー (C 単位)。

**limited_resource** を AP_YES または AP_NO_SESSIONS に設定すると、データがこのタイマーの設定 ~ 間内にリンクを通過せず、セッションがリンクを H 用しない場合、リンクは自動的に非 h 動されます。

**limited_resource** を AP_INACTIVITY に設定すると、データがこのタイマーの設定 ~ 間内にリンクを通過しない場合、リンクは自動的に非 h 動化されます。

**ls_detail.def_data.default_nn_server**
ネットワーク・ノード・サーバーとの CP-CP セッションをサポートするために、エンド・ノードによってリンクが自動的に h 動化されるかどうかを X 定します (AP_YES または AP_NO)。このフィールドの設定を有効にするためには、 CP_CP セッションをサポートするようにリンクを定義する , 要があります。

**ls_detail.def_data.ls_attributes**
隣接ノードに関するさらに詳しい情報を X 定します。

**ls_detail.def_data.ls_attributes[0]**
ホスト・タイプ。

**AP_SNA**
8 準 SNA ホスト。

**AP_FNA**
FNA (VTAM-F) ホスト。

**AP_HNA**
HNA ホスト。

**def_data.ls_attributes[1]**
このフィールドはビット・フィールドです。このフィールドには、AP_NO 値をX定するか、または以下の値をビット単位で OR で結合した任意の値をX定することができます。

**AP_SUPPRESS_CP_NAME**
バック・レベル LEN ノードとのリンクのためのネットワーク名 CV 抑 _ オプション。このビットを設定すると、ネットワーク名 CV が隣接ノードとの XID 交換に含められます。 (**adj_cp_type** が AP_BACK_LEVEL_LEN_NODE または AP_HOST_XID3 に設定されていない限り、このビットは無 k されます。)

**AP_REACTIVATE_ON_FAILURE**
リンクがアクティブになってから失敗すると、パーソナル・コミュニケーションズまたは Communications Serverはそのリンクを再度アクティブにしようと n みます。この n みに失敗すると、このリンクは非アクティブ状態を継続します。

**AP_USE_PU_NAME_IN_XID_CVS**
隣接ノードがホストとして定義されているか、または、 APPN とのリンクで、**solicit_sscp_sessions** が AP_YES に設定されている場合に、 AP_SUPPRESS_CP_NAME ビットが設定されていなければ、Format 3 XID で送信されたネットワーク名 CV の完全修飾 CP 名が、 **def_data.pu_name** で提供された名前 (CP のネットワーク ID で完全修飾されている) によって置換されます。

**ls_detail.def_data.adj_node_id**
隣接ノードの定義済みノード ID。

**ls_detail.def_data.local_node_id**
このリンク・ステーションの XID で送信されたノード ID。これは、4 バイトの 16 進数文 z 列です。このフィールドをゼロに設定すると、**node_id** が XID 交換でH用されます。このフィールドが非ゼロであれば、その値はこの LS の XID 交換の値と置換されます。

**ls_detail.def_data.cp_cp_sess_support**
CP-CP セッションがサポートされるかどうかをX定します (AP_YES または AP_NO)。

**ls_detail_def_data.use_default_tg_chars**
DEFINE_LS で提供された TG 特性が、 DEFINE_PORT で提供されたデフォルト特性のために廃棄されたかどうかをX定します (AP_YES または AP_NO)。このフィールドは、暗黙リンクには適用されません。

**ls_detail.def_data.tg_chars**
TG 特性 (33ページの『DEFINE_CN』 を 2 照)。

**ls_detail.def_data.target_pacing_count**
この TG の BIND 用として望ましいペーシング・ウィンドウを示す、 1 か

ら 32 767 までの数値。この数値は、固定バインド・ペーシングが実行される場合にのみ有効です。パーソナル・コミュニケーションズまたは Communications Serverは、現在この値をH用していませんので注意してください。

**ls_detail.def_data.max_send_btu_size**
送信可能な BTU の最大サイズ。

**ls_detail.def_data.ls_role**
このリンク・ステーションが果たすリンク・ステーション・ロール。このフィールドは、折衝可能、1 次、または 2 次を選択するための、AP_LS_NEG、AP_LS_PRI、または AP_LS_SEC のいずれかにすることができます。また、このフィールドを AP_USE_PORT_DEFAULTS に設定して、DEFINE_PORT verb で構成された値を選択することもできます。

**ls_detail.def_data.max_ifrm_rcvd**
肯定応答の前に XID 送信側が受信できる I フレームの最大数。DEFINE_PORT のデフォルト値をH用する , 要がある場合は、ゼロに設定してください。

**ls_detail.def_data.dlus_retry_timeout**
**ls_detail.def_data.dlus_name** および **ls_detail.def_data.bkup_dlus_name** フィールドにX定された DLUS とコンタクトするための、 2 番目の n 行とその後の n 行との間の間V (C単位)。最初の n 行と最初の再 n 行との間の間Vは、常に 1 Cです。ゼロをX定すると、DEFINE_DLUR_DEFAULTS を介して構成されたデフォルト値がH用されます。 **def_data.dspu_services** が AP_DLUR に設定されていなければ、このフィールドは無 k されます。

**ls_detail.def_data.dlus_retry_limit**
**ls_detail.def_data.dlus_name** および **ls_detail.def_data.bkup_dlus_name** フィールドにX定された DLUS とコンタクトするための最初の失敗の後、再 n 行を行った最大回数。ゼロをX定すると、DEFINE_DLUR_DEFAULTS を介して構成されたデフォルト値がH用されます。 X'FFFF' をX定すると、「プログラム」は無限に再 n 行を繰り返します。 **def_data.dspu_services** が AP_DLUR に設定されていなければ、このフィールドは無 k されます。

**ls_detail.def_data.link_spec_data_len**
初期設定中にリンク・ステーションの構成要素に未変更のまま渡されるデータの、埋め込みスペースを含まない長さ (バイト数)。このデータは、LS_DETAIL 構造体に連結されます。このデータは、 4 バイト境&の終わりまで埋め込まれています。

**ls_detail.def_data.convention_lu_compression**
このリンクでのセッションにデータ圧縮を要求するかどうかをX定します。このフィールドは、LU 0 ～ 3 のトラフィックを運ぶリンクにのみ有効であることに注意してください。

**AP_NO**
ローカル・ノードは、このリンクを流れる従来型の LU データを圧縮または解凍してはなりません。

**AP_YES**
ホストがデータ圧縮を要求した場合は、このリンクでの従来型の LU セッションについて、データ圧縮をH用可能にする，要があります。

**ls_detail.def_data.convention_lu_cryptography**
従来型の LU セッションにセッション・レベル暗号化が，要であるかどうかをX定します。このフィールドは、従来型の LU トラフィックを運ぶリンクの場合にのみ適用されます。

**AP_NONE**
「プログラム」は、セッション・レベル暗号化を行いません。

**AP_MANDATORY**
LU がインポート・キーをH用できる場合、「プログラム」は、，須のセッション・レベル暗号化を実行します。そうでない場合は、LUをH用するアプリケーションによってそれを実行しなければなりません (これが PU 集信であれば、それはダウンストリーム LU によって実行されます)。

**AP_OPTIONAL**
この値は、H用する暗号化が、セッションごとにホスト・アプリケーションによって起動されるようにします。ホストが、この PU に従属するセッションの暗号化を要求した場合は、「プログラム」は AP_MANDATORY の場合のような行動を取ります。ホストが暗号化を要求しなかった場合は、その行動は AP_NONE と同じになります。

**ls_detail.def_data.retry_flags**
このフィールドは、このリンク・ステーションのh動化が自動再n行される条件をX定します。これはビット・フィールドであり、以下の値をビット単位で OR で結合した任意の値を取ることができます。

**AP_RETRY_ON_START**
リンクのh動化をn行しているときにリモート・ノードから応答がないと、h動化が再n行されます。h動化をn行しているときに基本ポートが非アクティブ状態であると、「プログラム」はそれをh動化しようとします。

**AP_RETRY_ON_FAILURE**
リンクがアクティブまたは保留アクティブ状態のときに失敗すると、リンクのh動化が再n行されます。h動化をn行しているときに基本ポートが失敗すると、「プログラム」はそれをh動化しようとします。

**AP_RETRY_ON_DISCONNECT**
リンクがリモート・ノードによって正常停 _ されると、リンクのh動化が再n行されます。

**AP_DELAY_APPLICATION_RETRIES**
アプリケーションによって+Oされた (START_LS またはオンデマンド・リンクh動化をH用して) リンクh動化再n行は、**activation_delay_timer** をH用して歩調合わせされます。

**AP_DELAY_INHERIT_RETRY**
このフィールドのフラグでX定された再 n 行条件のほかに、基本ポート定義の **retry_flags** フィールドにX定された再 n 行条件もH用されます。

**ls_detail.def_data.max_activation_attempts**
少なくとも 1 つのフラグが **retry_flags** に設定されない限り、このフィールドは効果を生じません。

このフィールドは、リモート・ノードが無応答の場合、または基本ポートが非 h 動状態の場合に「プログラム」によって許容される再 n 行の回数をX定します。この回数には、自動再 n 行とアプリケーション主導型の h 動化 n 行の両方の回数も含まれます。

この限度に達すると、自動再 n 行はこれ以上行われません。この条件は、STOP_LS、STOP_PORT、STOP_DLC、または成功した h 動化によってリセットされます。 START_LS または OPEN_LU_SSCP_SEC_RQ によって 1 回の h 動化 n 行が行われますが、 h 動化に失敗すると、再 n 行は行われません。

ゼロは ’限度がない’ ことを意味します。 AP_USE_DEFAULTS の値をX定すると、 DEFINE_PORT で提供された **max_activiation_attempts** がH用されます。

**ls_detail.def_data.activation_delay_timer**
少なくとも 1 つのフラグが **retry_flags** に設定されない限り、このフィールドは効果を生じません。

このフィールドは、 AP_DELAY_APPLICATION_RETRIES ビットが **def_data.retry_flags** に設定されている場合に、「プログラム」が自動再 n 行間、およびアプリケーション主導型 h 動化 n 行間に待機するC数をX定します。

AP_USE_DEFAULTS の値をX定すると、 DEFINE_PORT で提供された **activiation_delay_timer** がH用されます。

ゼロをX定すると、「プログラム」は 30 C間、デフォルトのタイマーをH用します。

**def_data.branch_link_type**
BrNN のみ。リンクがアップリンクであるか、ダウンリンクであるかをX定します。このフィールドは、**def_data.adj_cp_type** フィールドが、AP_NETWORK、NODE、AP_END_NODE、AP_APPN_NODE、または AP_BACK_LEVEL_LEN_NODE に設定されている場合にのみ適用されます。

**AP_UPLINK**
このリンクはアップリンクです。

**AP_DOWNLINK**
このリンクはダウンリンクです。

**adj_cp_type** フィールドを AP_NETWORK_NODE に設定した場合は、このフィールドを AP_UPLINK に設定する , 要があります。

その他のノード・タイプ: このフィールドは無 k されます。

**ls_detail.det_data.adj_cp_is_brnn**

BrNN のみ。隣接 CP が、たとえば、BrNN がその NN フェースを示す NN(BrNN) であることが許可されているか、 , 要であるか、あるいは禁 _ されているかを X 定します。このフィールドは、 **adj_cp_type** フィールドが AP_NETWORK_NODE または AP_APPN_NODE に設定され (かつ、 XID 交換 ~ に N 認された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。

**AP_BRNN_ALLOWED**

隣接 CP が NN(BrNN) であることが許可されています (ただし、 , 要ではありません)。

**AP_BRNN_REQUIRED**

隣接 CP が NN(BrNN) であることが許可されていません。

**AP_BRNN_PROHIBITED**

隣接 CP が NN(BrNN) であることが許可されていません。

**adj_cp_type** フィールドを AP_NETWORK_NODE に設定し、 **auto_act_supp** フィールドを AP_YES に設定した場合は、このフィールドを AP_BRNN_REQUIRED または AP_BRNN_PROHIBITED に設定する , 要があります。

その他のノード・タイプ: このフィールドは無 k されます。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_LINK_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_LS_EXCEPTION

QUERY_LS は、ノードで定義されたリンク・ステーションに関する情報のリストを戻します。この情報は、『決定済みデータ』 (実行中に動的に収集されたデータ) および『定義済みデータ』 (DEFINE_LS のアプリケーションによって提供されたデータ) として構造化されます。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の LS に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **ls_name** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**ls_name** 順に配列されます。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

戻されたリンク・ステーションのリストは、自分が属するポートの名前別にフィルター処理することができます。この場合は、**port_name** フィールドを設定する , 要があります (それ以 0 の場合は、このフィールドをすべてゼロに設定する , 要があります)。

## VCB 構造体

```
typedef struct query_ls_exception
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* Primary return code            */
        unsigned long   secondary_rc;        /* Secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  total_num_entries;   /* total number of entries        */
        unsigned char   list_options;        /* listing options                */
        unsigned char   reserv3;             /* reserved                       */
        unsigned long   exception_index;     /* index of LS exception entry    */
        unsigned char   ls_name;             /* name of link station           */
} QUERY_LS_EXCEPTION;

typedef struct LS_EXCEPTION
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned long   exception_indes;     /* index of this entry            */
        unsigned_DATE_TIME  time;            /* date and time                  */
        unsigned char   ls_name[8];          /* link station name              */
        unsigned char   adj_cp_name[17];     /* adjacent CP name               */
        unsigned char   adj_node_id[4];      /* adjacent node id               */
        unsigned short  tg_number;           /* TG number                      */
        unsigned long   general_sense;       /* general sense data             */
        unsigned char   retry;               /* wil retry request              */
        unsigned long   end_sense;           /* termination sense data         */
```

```
        unsigned long   xid_local_sense;     /* XID local sense data          */
        unsigned long   xid_remote_sense;    /* XID remote sense data         */
        unsigned short  xid_error_byte;      /* offset of byte in error       */
        unsigned short  xid_error_bit;       /* offset of bit in error        */
        unsigned char   dlc_type;            /* DLC type                      */
        LINK_ADDRESS    local_addr;          /* local address                 */
        LINK_ADDRESS    destination_addr;    /* destination address           */
        unsigned char   reserved[20];        /* reserved                      */
} LS_EXCEPTION;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**
AP_QUERY_LS_EXCEPTION

**format**
VCB の 形式を識別します。上記リストの VCB の Format 1 バージョンをX定するには、このフィールドを 1 に設定します。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
これは、リスト情報に何を戻すかを示します。

以下のパラメーターにX定された **index** は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、入力した索引値が示す項目の次の項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**exception_index**
LS 例 0 項目の索引。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**ls_name**
戻された項目が関連するリンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。こ

のフィールドをヌルに設定すると、任意のリンク・ステーションまたはすべてのリンク・ステーションに関連する項目が戻されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**ls_exception.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**ls_exception.exception_index**
この LS 例 0 項目に d り当てられた索引。索引値は、ゼロでO まり、最大値 2**31-1 (2,147,483,647) まで増加してから折り返されます。

**ls_exception.time**
LS 例 0 項目が生成された日U と ~ 刻。

**ls_exception.ls_name**
リンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**ls_exception.adj_cp_name**
17 バイトの完全修飾隣接 CP 名。1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) このフィールドの値は、次のようにして決定されます。

XID で隣接 CP 名を受信すると、それは戻されます。

XID で隣接 CP 名を受信したが、ローカルに定義された値がH用できれば、それは戻されます。

それ以 0 の場合は、ヌルが戻されます。

**ls_exception.node_id**
XID 交換 ~ に隣接ノードから受信したノード ID (何も受信しなかった場合は、ヌル)。これは、4 バイトの 16 進数文 z 列です。

## QUERY_LS_EXCEPTION

**ls_exception.tg_number**
このリンク・ステーションとの TG と関連する番号。0 から 256 までの範囲です。 256 の値は、障 2 ~ に TG 番号が認識されなかったことを示します。

**ls_exception.general_sense**
XID 順序 + O までの、リンク h 動化 + O 順序に関連するエラー・センス・データ。これはノードによって生成されます。

**ls_exception.retry**
ノードがリンク h 動化の + O 要求を再 n 行するかどうかを示します。

**AP_NO**
ノードは + O 要求を再 n 行しません。

**AP_YES**
ノードは + O 要求を再 n 行します。

**ls_exception.end_sense**
h 動化 n 行の終了に関連するセンス・データ。これはデータ・リンク制御層によって生成されます。

**ls_exception.xid_local_sense**
XID で送信されたローカル生成センス・データ。

**ls_exception.xid.remote_sense**
XID で受信されたリモート生成センス・データ。

**ls_exception.xid_error_byte**
XID のエラー・バイトに含まれているエラー・ビットのオフセット (0 から 65535 まで)。 65535 の値は、このフィールドが意味をもっていないことを示します。

**ls_exception.xid_error_bit**
XID のエラー・バイトに含まれているエラー・ビットのオフセット (0 から 7 まで)。 8 の値は、このフィールドが意味をもっていないことを示します。

**ls_exception.dlc_type**
DLC のタイプ。パーソナル・コミュニケーションズまたは Communications Serverは、以下のタイプをサポートします。

AP_SDLC
AP_X25
AP_TR

DEFINE_DLC verb に新しいタイプをX定することによって、追加の DLC タイプを定義することができます。詳細については、 49ページの『DEFINE_DLC』 を2照してください。

**ls_exception.local_addr.length**
ローカル・リンク・ステーションのアドレスの長さ。

**ls_exception.local_address.address**
ローカル・リンク・ステーションのアドレス。

**ls_exception.destination_addr.length**
隣接ノードでの宛先リンク・ステーションのアドレスの長さ。

**ls_exception.destination_addr.address**
隣接ノードでの宛先リンク・ステーションのアドレス。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_EXCEPTION_INDEX

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_LU_0_TO_3

QUERY_LU_0_TO_3 は、タイプ 0、1、2、3 のローカル LU に関する情報を戻します。この情報は、『決定済みデータ』 (実行中に動的に収集されたデータ) および『定義済みデータ』 (DEFINE_LU 0 TO 3 のアプリケーションによって提供されたデータ) として構造化されます。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のローカル LU に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **lu_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。

SNA API クライアントでは、特定のパラメーターのみがサポートされます。特定の詳細情報については、*メモ帳* を 2 照してください。

このアイコンは、Communications Serverおよび パーソナル・コミュニケーションズ の操作に影響を与える可能性のある重要な情報を示しています。

## VCB 構造体

```
typedef struct query_lu_0_to_3
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* Verb attributes               */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   pu_name[8];          /* PU name                       */
        unsigned char   lu_name[8];          /* LU name                       */
        unsigned char   description[RD_LEN]; /* resource description          */
        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
{
        unsigned short  overlay_size;        /* size of this 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;       /* LU-SSCP session statistics    */
        SESSION_STATS   plu_stats;           /* PLU-SLU session statistics    */
        unsigned char   plu_name[8];         /* PLU name                      */
        unsigned char   session_id[8];       /* Internal ID of PLU-SLU sess   */
        unsigned char   app_spec_det_data[256];
                                             /* Application Specified Data    */
        unsigned char   app_type;            /* Application type              */
        unsigned char   sscp_id[6];          /* SSCP ID                       */
        unsigned char   bind_lu_type;        /* LU type issuing BIND          */
        unsigned char   reserva[12];         /* reserved                      */
} LU_0_TO_3_DET_DATA;

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size       */
        unsigned short  send_ru_size;        /* session send RU size          */
        unsigned short  max_send_btu_size;   /* max send BTU size             */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size              */
        unsigned short  max_send_pac_win;    /* max send pacing win size      */
        unsigned short  cur_send_pac_win;    /* current send pacing win size  */
        unsigned short  max_rcv_pac_win;     /* max receive pacing win size   */
        unsigned short  cur_rcv_pac_win;     /* current receive pacing        */
                                             /* window size                   */
        unsigned long   send_data_frames;    /* number of data frames sent    */
        unsigned long   send_fmd_data_frames;
                                             /* num of FMD data frames sent   */
        unsigned long   send_data_bytes;     /* number of data bytes sent     */
        unsigned long   rcv_data_frames;     /* num data frames received      */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames recvd  */
        unsigned long   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[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[RD_LEN]; /* resource description          */
        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                      */
        unsigned char   sscp_id[6];
        unsigned char   timeout;             /* Timeout                       */
        unsigned char   app_spec_def_data[16];
                                             /* Application Specified Data    */
        unsigned char   model_name[7];       /* LU model                      */
        unsigned char   reserv3[17];         /* reserved                      */
} LU_0_TO_3_DEF_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

## QUERY_LU_0_TO_3

**opcode**

AP_QUERY_LU_0_TO_3

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーを X すポインター。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

これは、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

AP_SUMMARY 値は、SNA API クライアントについてもサポートされます。

**AP_DETAIL**

詳細情報を戻します。

X 定された **lu_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O 点を X 定するために H 用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

AP_FIRST_IN_LIST 値は、SNA API クライアントについてもサポートされます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によって X 定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から + O されます。

**lu_name**

照会されるローカル LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

SNA API クライアントの場合、list_options 値は無 k されます。

**pu_name**

PU 名。この PU を H 用する LU のみが戻されます。すべての LU のリストが , 要であれば、このフィールドをすべて 2 進ゼロに設定する , 要があります。 SNA API クライアントの場合、list_options 値は無 k されます。

SNA API クライアントの場合、pu_name 値は無 k されます。

**host_attachment**

ホスト処理装置接続機構のためのフィルター。

**AP_NONE**

すべての LU に関する情報を戻します。

AP_NONE は、SNA API クライアントの場合にサポートされている唯一の値です。

**AP_DLUR_ATTACHED**

DLUR によってサポートされているすべての LU に関する情報を戻します。

**AP_DIRECT_ATTACHED**

ホスト・システムに直接接続されている LU に関する情報のみを戻します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**

実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**lu_0_to_3_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**lu_0_to_3_summary.pu_name**
この LU がH用するローカル PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

SNA API クライアントでは、lu_0_to_3_summary.pu_name 値は戻されません。

**lu_0_to_3_summary.lu_name**
照会されるローカル LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**lu_0_to_3_summary.description**
q 源の説明 (DEFINE_LU_0_TO_3 でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

SNA API クライアントでは、lu_0_to_3_summary.description 値は戻されません。

**lu_0_to_3_summary.nau_address**
LU のネットワーク・アドレス単位アドレス。範囲は 1 〜 255。

SNA API クライアントでは、lu_0_to_3_summary.nau_address 値は戻されません。

**lu_0_to_3_summary.lu_sscp_sess_active**
LU-SSCP セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

SNA API クライアントでは、lu_0_to_3_summary.lu_sscp_sess_active 値は戻されません。

**lu_0_to_3_summary.appl_conn_active**
アプリケーションが LU をH用するかどうかを示します (AP_YES または AP_NO)。

SNA API クライアントでは、lu_0_to_3_summary.aapl_conn_active 値は戻されません。

**lu_0_to_3_summary.plu_sess_active**
PLU-SLU セッションがアクティブであるかどうかを示します (AP_YES また

は AP_NO)。

SNA API クライアントでは、lu_0_to_3_summary.plu_sess_active 値は戻されません。

**lu_0_to_3_summary.host_attachment**
LU ホスト処理装置接続機構タイプ。

**AP_DLUR_ATTACHED**
LU は、DLUR によってホスト・システムと接続されています。

**AP_DIRECT_ATTACHED**
LU はホスト・システムと直接接続しています。

**lu_0_to_3_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**lu_0_to_3_detail.lu_name**
照会されるローカル LU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。

**lu_0_to_3_detail.det_data.lu_sscp_sess_active**
LU-SSCP セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

**lu_0_to_3_detail.det_data.appl_conn_active**
この LU が現在アプリケーションによってH用されているかどうかを示します (AP_YES または AP_NO)。

**lu_0_to_3_detail.det_data.plu_sess_active**
PLU-SLU セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

**lu_0_to_3_detail.det_data.host_attachment**
LU ホスト処理装置接続機構タイプ。

**AP_DLUR_ATTACHED**
LU は、DLUR によってホスト・システムと接続されています。

**AP_DIRECT_ATTACHED**
LU はホスト・システムと直接接続しています。

**lu_0_to_3_detail.det_data.lu_sscp_stats.rcv_ru_size**
このフィールドは常に予約済みです。

**lu_0_to_3_detail.det_data.lu_sscp_stats.send_ru_size**
このフィールドは常に予約済みです。

**lu_0_to_3_detail.det_data.lu_sscp_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**lu_0_to_3_detail.det_data.lu_sscp_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**lu_0_to_3_detail.det_data.lu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_0_to_3_detail.det_data.lu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_0_to_3_detail.det_data.lu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_0_to_3_detail.det_data.lu_sscp_stats.cur_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_0_to_3_detail.det_data.lu_sscp_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**lu_0_to_3_detail.det_data.lu_sscp_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**lu_0_to_3_detail.det_data.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 _0_to_3_detail.det_data.lu_sscp_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**lu_0_to_3_detail.det_data.lu_sscp_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**lu_0_to_3_detail.det_data.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_0_to_3_detail.det_data.lu_sscp_stats.sidh**
セッション ID 上位バイト。

**lu_0_to_3_detail.det_data.lu_sscp_stats.sidl**
セッション ID 下位バイト。

**lu_0_to_3_detail.det_data.lu_sscp_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、このフィールドをゼロに設定し、ACTLU 送信側に 2 次リンク・ステーションが含まれていれば、このフィールドを 1 に設定します。

**lu_0_to_3_detail.det_data.lu_sscp_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。このフィールドを H 用すれば、このセッションと、このセッションが通るリンクとを相関させることができます。

**lu_0_to_3_detail.det_data.lu_sscp_stats.pacing_type**
LU-SSCP セッションで H 用される受信ペーシング・タイプ。これは AP_NONE に設定されます。

**lu_0_to_3_detail.det_data.plu_stats.rcv_ru_size**
受信 RU の最大サイズ。

**lu_0_to_3_detail.det_data.plu_stats.send_ru_size**
送信 RU の最大サイズ。

**lu_0_to_3_detail.det_data.plu_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**lu_0_to_3_detail.det_data.plu_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**lu_0_to_3_detail.det_data.plu_stats.max_send_pac_win**
このセッションでの送信ペーシング・ウィンドウの最大サイズ。

**lu_0_to_3_detail.det_data.plu_stats.cur_send_pac_win**
このセッションでの送信ペーシング・ウィンドウの現行サイズ。

**lu_0_to_3_detail.det_data.plu_stats.max_rcv_pac_win**
このセッションでの受信ペーシング・ウィンドウの最大サイズ。

**lu_0_to_3_detail.det_data.plu_stats.cur_rcv_pac_win**
このセッションでの受信ペーシング・ウィンドウの現行サイズ。

**lu_0_to_3_detail.det_data.plu_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**lu_0_to_3_detail.det_data.plu_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**lu_0_to_3_detail.det_data.plu_stats.send_data_bytes**
送信された通常フロー・データ・バイトの数。

**lu_0_to_3_detail.det_data.plu_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**lu_0_to_3_detail.det_data.plu_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**lu_0_to_3_detail.det_data.plu_stats.rcv_data_bytes**
受信された通常フロー・データ・バイトの数。

**lu_0_to_3_detail.det_data.plu_stats.sidh**
セッション ID 上位バイト。

**lu_0_to_3_detail.det_data.plu_stats.sidl**
セッション ID 下位バイト。

**lu_0_to_3_detail.det_data.plu_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、BIND の送信側はこのフィールドをゼロに設定し、 BIND の送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドを 1 に設定します。

**lu_0_to_3_detail.det_data.plu_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**lu_0_to_3_detail.det_data.plu_stats.pacing_type**
PLU-SSCP セッションでH用される受信ペーシング・タイプ。このフィールドには、AP_NONE または AP_PACING_FIXED 値を入れることができます。

**lu_0_to_3_detail.det_data.plu_name**
1 次 LU 名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。(PLU-SLU セッションが非アクティブであれば、このフィールドは予約済みになります)。

**lu_0_to_3_detail.det_data.session_id**
PLU_SLU セッションの 8 バイトの内 t ID。

**lu_0_to_3_detail.det_data.app_spec_det_data**
予約済み。

**lu_0_to_3_detail.det_data.sscp_id**
これは 6 バイトのフィールドで、このフィールドには、この LU でH用された PU の ACTPU で受信された SSCP ID が含まれています。

**lu_sscp_sess_active** が AP_YES でなければ、このフィールドはゼロになります。

**lu_0_to_3_detail.det_data.app_type**
予約済み。

**lu_0_to_3_detail.lu_0_to_3_det_data.bind_lu_type**
オリジナルの BIND を発行した LU の LU タイプ。アクティブ LU-LU セッションがあれば、このフィールドは以下のいずれかになります。

AP_LU_TYPE_0
AP_LU_TYPE_1
AP_LU_TYPE_2
AP_LU_TYPE_3
AP_LU_TYPE_6 (ダウンストリーム従属 LU 6.2 の場合)

アクティブ LU−LU セッションがなければ、このフィールドは以下の値を取ります。

AP_LU_TYPE_UNKNOWN

**lu_0_to_3_detail.def_data.description**
q 源の説明 (DEFINE_LU_0_TO_3 でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**lu_0_to_3_detail.def_data.nau_address**
LU のネットワーク・アドレス単位アドレス。範囲は 1 〜 255。

**lu_0_to_3_detail.def_data.pool_name**
この LU が属するプールの名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。この LU が特定のプールに属していない場合、このフィールドはすべて 2 進ゼロに設定されます。

**lu_0_to_3_detail.def_data.pu_name**
この LU がH用する PU の名前 (DEFINE_LS verb でX定)。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**lu_0_to_3_detail.def_data.priority**
ホストへの送信 ~ の LU の優先順位。以下のいずれかの値に設定されます。

AP_NETWORK
AP_HIGH
AP_MEDIUM
AP_LOW

**lu_0_to_3_detail.def_data.lu_model**
LU のモデル・タイプと番号。以下のいずれかの値に設定されます。

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

**lu_0_to_3_detail.def_data.sscp_id**
このフィールドは、この LU を h 動化することが許可された SSCP の ID を X定します。これは 6 バイトの 2 進数フィールドです。このフィールドを 2 進ゼロに設定すると、任意の SSCP によって LU が h 動化されます。

**lu_0_to_3_detail.def_data.timeout**
X定された LU のタイムアウト (C 単位)。タイムアウトがX定されていて、LU のユーザーが OPEN_LU_SSCP_SEC_RQ に (または PU 集信の場合は、ダウンストリーム LU 定義に) **allow_timeout** をX定している場合は、 LU を非 h 動化する前に、PLU-SLU セッションをこの期間非 h 動にしておき、以下のいずれかの条件を保留します。

- セッションが限定 q 源リンクを通過する
- セッションが再H用される前に、別のアプリケーションが LU をH用する

タイムアウトをゼロに設定すると、LU は非 h 動化されません。

**lu_0_to_3_detail.def_data.app_spec_def_data**
DEFINE_LU_0_TO_3 からのアプリケーションX定のデータ。「プログラム」はこのフィールドを解釈しません。このデータは、単に、QUERY_LU_0_TO_3 verb で J 納され、戻されるだけです。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**QUERY_LU_0_TO_3**

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_LU_POOL

QUERY_LU_POOL は、プールのリストおよびプールに属する LU のリストを戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の LU プールに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **pool_name** および **lu_name** フィールドを設定する , 要があります。 **lu_name** フィールドをすべてゼロに設定すると、戻された情報はX定されたプール内の最初の LU から + O されます。 **list_options** フィールドを AP_FIRST_IN_LIST に設定すると、このどちらのフィールドも無 k されます。

## VCB 構造体

```
typedef struct query_lu_pool
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   pool_name[8];        /* pool name                     */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned short  num_active_lus;      /* num of currently active LUs   */
        unsigned char   num_avail_lus;       /* num of currently available    */
                                             /* LUs                           */
} LU_POOL_SUMMARY;

typedef struct lu_pool_detail
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   pool_name[8];        /* pool name                     */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   lu_name[8];          /* LU name                       */
        unsigned char   lu_sscp_sess_active; /* Is LU-SSCP session active     */
        unsigned char   appl_conn_active;    /* Is SSCP connection open       */
        unsigned char   plu_sess_active;     /* Is PLU-SLU session active     */
} LU_POOL_DETAIL;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_LU_POOL

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンを X 定義 ?f 0.9872

べて 2 進ゼロに設定すると、X定されたプールに属する LU は、プールの先頭からリストされます。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
戻されたディレクトリー項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**lu_pool_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**lu_pool_summary.pool_name**
X定された LU が属する LU プールの名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。(このフィールドを要求でX定し、 **lu_name** フィールドをすべて 2 進ゼロに設定すると、プール内の LU のみが戻されますので注意してください。)

**lu_pool_summary.description**
LU プール記述 (DEFINE_LU_POOL でX定)。

**lu_pool_summary.num_active_lus**
アクティブ LU-SSCP セッションをもつ、X定プール内の LU の数。

**lu_pool_summary.num_avail_lus**
**open_force** を AP_YES に設定して OPEN_LU_SSCP_SEC_REQ を満たすためにH用できる、X定プール内の LU の数。これには、PU がアクティブ状態になっているすべての LU、またはホスト・リンクが自動的にアクティブ状態になるすべての LU、および接続が空き状態になっているすべての LU が含まれます。この数は、 LU の **model_type**、**model_name**、およびPU の DDDLU サポートとは関係ありません。特定の値を **model_type** にX定する OPEN_LU_SSCP_SEC_REQ を満たすためにH用できる LU が少なくなることがあります。

**lu_pool_detail.num_active_lus**
アクティブ LU-SSCP セッションをもつ、X定プール内の LU の数。

**lu_pool_detail.num_avail_lus**
H用可能な LU-SSCP セッションをもつ、X定プール内の LU の数。

**lu_pool_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**lu_pool_detail.pool_name**
X定された LU が属する LU プールの名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。(このフィールドを要求でX定し、 **lu_name** フィールドをすべて 2 進ゼロに設定すると、プール内の LU のみが戻されますので注意してください。)

**lu_pool_detail.description**
LU 記述 (DEFINE_LU_0_TO_3 でX定)。

**lu_pool_detail.lu_name**
このプールに属している LU の LU 名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。この名前がすべてゼロに設定されている場合は、X定されたプールが空であることを示します。

**lu_pool_detail.lu_sscp_sess_active**
LU-SSCP セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

**lu_pool_detail.appl_conn_active**
この LU セッションが現在アプリケーションによってH用されているかどうかを示します (AP_YES または AP_NO)。

**lu_pool_detail.plu_sess_active**
PLU-SLU セッションがアクティブであるかどうかを示します (AP_YES または AP_NO)。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LIST_OPTION

AP_INVALID_POOL_NAME
AP_INVALID_LU_NAME

ノードが+ Oされないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_MDS_APPLICATION

QUERY_MDS_APPLICATION は、MDS レベル・メッセージ用に登録したアプリケーションのリストを戻します。

アプリケーションを登録するには、 675ページの『第15章 管理サービス verb』に説明されている REGISTER_MS_APPLICATION verb をH用します。

特定のアプリケーションに関する情報または 『固まり』 に分けられたリスト情報を入手するには、 **application** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

## VCB 構造体

```
typedef struct query_mds_application
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   application[8];      /* application name              */
        unsigned short  max_rcv_size;        /* max data size application     */
                                             /* can receive                   */
        unsigned char   reserva[20];         /* reserved                      */
} MDS_APPLICATION_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_MDS_APPLICATION

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
これは、リスト情報に何を戻すかを示します。つまり、X定された **application** (以下のパラメーターを2照) は、戻された実際の情報の+O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無kされ、戻りリストはリスト内の最初の項目から+Oされます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から+Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から+Oされます。

**application**
アプリケーション名。この名前は、8 バイト英数 z のタイプ A の EBCDIC 文 z 列です。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**mds_application_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**mds_application_data.application**
登録されたアプリケーションの名前。この名前は、8 バイト英数 z のタイプ A の EBCDIC 文 z 列です。

**mds_application_data.max_rcv_size**
アプリケーションが 1 つの固まりとして受信することができるバイトの最大数 (この値は、アプリケーションが MDS に登録するときにX定されます)。MDS レベルのアプリケーション登録の詳細については、 第15章 管理サービス verb を 2 照してください。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_APPLICATION_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_MDS_STATISTICS

QUERY_MDS_STATISTICS は、管理サービス統計を戻します。この verb をH用して MDS 経路X定トラフィックのレベルを測定することができます。

## VCB 構造体

```
typedef struct query_mds_statistics
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned long   alerts_sent;         /* number of alert sends         */
        unsigned long   alert_errors_rcvd;   /* error messages received       */
                                             /* for alert sends               */
        unsigned long   uncorrelated_alert_errors;
                                             /* uncorrelated alert            */
                                             /* errors received               */
        unsigned long   mds_mus_rcvd_local;  /* number of MDS_MUs received    */
                                             /* from local applications       */
        unsigned long   mds_mus_rcvd_remote; /* number of MDS_MUs received    */
                                             /* from remote applications      */
        unsigned long   mds_mus_delivered_local;
                                             /* num of MDS_MUs delivered      */
                                             /* to local applications         */
        unsigned long   mds_mus_delivered_remote;
                                             /* num of MDS_MUs                */
                                             /* delivered to remote appls     */
        unsigned long   parse_errors;        /* number of MDS_MUs received    */
                                             /* with parse errors             */
        unsigned long   failed_deliveries;   /* number of MDS_MUs where       */
                                             /* delivery failed               */
        unsigned long   ds_searches_performed;
                                             /* number of DS searches done    */
        unsigned long   unverified_errors;   /* number of unverified errors   */
        unsigned char   reserva[20];         /* reserved                      */
} QUERY_MDS_STATISTICS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_MDS_STATISTICS

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

**alerts_sent**

MDS 移送システムをH用して送信されたローカル発信アラートの数。

## QUERY_MDS_STATISTICS

**alert_errors_rcvd**
アラートが含まれたメッセージの送達障 2 を示すエラー・メッセージを MDS が受信した回数。

**uncorrelated_errors_rcvd**
アラートが含まれたメッセージの送達障 2 を示すエラー・メッセージを MDS が受信した回数。送達障 2 は、エラー・メッセージを MDS 送信アラート待ち行列のアラートと相関できなかったときに発生します。 MDS は、問題判別フォーカル・ポイントに送信されたアラートをキャッシュするための固定サイズ待ち行列を維 } します。待ち行列が最大サイズに達すると、 1 番古いアラートが破棄され、新しいアラートと置換されます。送達エラー・メッセージが受信すると、MDS は、エラー・メッセージをキャッシュ・アラートと相関させて、問題判別フォーカル・ポイントが | 元されるまでアラートを保 } できるように n みます。

**注:** **alert_errors_rcvd** と **uncorrelated_errors_rcvd** の 2 つの数は、送信アラート待ち行列のサイズを調整できるように維 } されます。 ~ 間の経過につれて **uncorrelated_errors_rcvd** が増える場合は、送信アラート待ち行列サイズが小さすぎることを示しています。

**mds_mus_rcvd_local**
ローカル・アプリケーションから受信された MDS_MU の数。

**mds_mus_rcvd_remote**
MDS_RECEIVE および MSU_HANDLER トランザクション・プログラムを H 用して、リモート・ノードから受信された MDS_MU の数。

**mds_mus_delivered_local**
ローカル・アプリケーションに正常に送達された MDS_MU の数。

**mds_mus_delivered_remote**
MDS_SEND トランザクション・プログラムを H 用して、リモート・ノードに正常に送達された MDS_MU の数。

**parse_errors**
受信された、ヘッダー形式エラーを含む MDS_MU の数。

**failed_deliveries**
このノードが送達することに失敗した MDS_MU の数。

**ds_searches_performed**
予約済み。

**unverified_errors**
予約済み。

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_MODE

QUERY_MODE は、特定のパートナー LU をもつローカル LU によってH用されているモードに関する情報を戻します。この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のモードに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **mode_name** フィールドを設定する , 要があります。そうしないと (**list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。**lu_name** (または **lu_alias**) および **plu_alias** (または **fqplu_name**) フィールドを、常に設定しておく , 要があることに注意してください。**lu_name** が非ゼロであれば、**lu_alias** よりも優先してH用されます。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストには、**lu_name** (または **lu_alias**) によってX定されたローカル LU に関する情報のみが含まれています。このリストは、まず **fqplu_name** 順に、次に **mode_name** 順に配列されます。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。

**plu_alias** をすべてゼロに設定すると、 **fqplu_name** 値がH用されます。そうでない場合は、 **plu_alias** が常にH用され、 **fqplu_name** は無 k されます。

戻されたモードのリストは、これらのモードが現在アクティブ・セッションをもっているかどうかに基づいてフィルター処理されます。フィルター処理を行う場合は、 **active_sessions** フィールドを AP_YES に設定する , 要があります (フィルター処理を行わない場合は、このフィールドを AP_NO に設定する , 要があります)。この verb は、パートナー LU をもつローカル LU がこのモードをH用しOめるときに決定された情報を戻します。 QUERY_MODE_DEFINITION verb は、定義情報のみを戻します。

## VCB 構造体

```
typedef struct query_mode
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   *buf_ptr;           /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   mode_name[8];        /* mode name                     */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned short  sess_limit;          /* current session limit         */
        unsigned short  act_sess_count;      /* curr active sessions count    */
        unsigned char   fqplu_name[17];      /* partner LU name               */
        unsigned char   reserv1[3];          /* reserved                      */
} MODE_SUMMARY;

typedef struct mode_detail
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   mode_name[8];        /* mode name                     */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned short  sess_limit;          /* session limit                 */
        unsigned short  act_sess_count;      /* currently active sess count   */
        unsigned char   fqplu_name[17];      /* partner LU name               */
        unsigned char   reserv1[3];          /* reserved                      */
        unsigned short  min_conwinners_source;
                                             /* min conwinner sess limit      */
        unsigned short  min_conwinners_target;
                                             /* min conloser limit            */
        unsigned char   drain_source;        /* drain source?                 */
        unsigned char   drain_partner;       /* drain partner?                */
        unsigned short  auto_act;            /* auto activated conwinner      */
                                             /* session limit                 */
        unsigned short  act_cw_count;        /* active conwinner sess count   */
        unsigned short  act_cl_count;        /* active conloser sess count    */
        unsigned char   sync_level;          /* synchronization level         */
        unsigned char   default_ru_size;     /* default RU size to maximize   */
                                             /* performance                   */
        unsigned short  max_neg_sess_limit;  /* max negotiated session limit  */
        unsigned short  max_rcv_ru_size;     /* max receive RU size           */
        unsigned short  pending_session_count;
                                             /* pending sess count for mode   */
        unsigned short  termination_count;   /* termination count for mode    */
        unsigned char   implicit;            /* implicit or explicit entry    */
        unsigned char   reserva[15];         /* reserved                      */
} MODE_DETAIL;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_MODE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

## QUERY_MODE

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

**lu_name** (または、 **lu_name** がすべてゼロに設定されている場合は、**lu_alias**)、 **plu_alias** (または、**plu-alias** がすべてゼロに設定されている場合は、 **fqplu_name**)、および **mode_name** を組み合わせたX定 (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。あるパートナー LU の索引をX定すると、可能な場合、他のパートナー LU に関する情報がこのリストに組み込まれます。

**AP_FIRST_IN_LIST**
**plu_alias** および **fqplu_name** をすべてゼロに設定すると、戻りリストはリスト内の最初のパートナー LU から + O され、 **mode_name** 索引は無 k されます。 **plu_alias** または **fqplu_name** のいずれかをX定すると、リストはこの索引から + O されますが、**mode_name** 索引値は無 k され、戻りリストがリスト内の最初のモード項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**lu_name**
LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。このフィールドをすべてゼロに設定すると、 **lu_alias** フィールドが索引の判別にH用されます。

**lu_alias**
ローカル定義の LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。このフィールドは、**lu_name** フィールドをすべてゼロに設定した場合にのみ有効です。この場合、8 バイトすべてが有効であるため、8 バイトすべてを設定する , 要があります。**lu_name** と **lu_alias** を両方ともすべてゼロに設定すると、制御点と関連する LU (デフォルトの LU) がH用されます。

**plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。このフィールドをすべてゼロに設定すると、 **fqplu_name** フィールドが索引値の判別にH用されます。

**fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**mode_name**
モード名。これは、セッション・グループのネットワーク特性を X 定します。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**active_sessions**
アクティブ・セッション・フィルター。戻されたモードを、それらが現在アクティブ・セッションをもっているかどうかに基づいてフィルター処理を行うかどうかを X 定します (AP_YES または AP_NO)。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**mode_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**mode_summary.mode_name**
モード名。これは、セッション・グループのネットワーク特性を X 定します。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。

**mode_summary.description**
q 源の説明 (DEFINE_MODE で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**mode_summary.sess_limit**
現行のセッション限度。

**mode_summary.act_sess_count**
モードをH用するアクティブ・セッションの合計数。**active_sessions** フィルターが AP_YES に設定されていれば、このフィールドは常にゼロより大きくなります。

**mode_summary.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文z列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**mode_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**mode_detail.mode_name**
モード名。これは、セッション・グループのネットワーク特性をX定します。これは、8 バイト英数 z のタイプ A の EBCDIC 文z列 (文z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**mode_detail.description**
q 源の説明 (DEFINE_MODE でX定)。

**mode_detail.sess_limit**
現行のセッション限度。

**mode_detail.act_sess_count**
モードをH用するアクティブ・セッションの合計数。**active_sessions** フィルターが AP_YES に設定されていれば、このフィールドは常にゼロより大きくなります。

**mode_detail.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文z列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**mode_detail.min_conwinners_source**
ローカル LU が競合勝者 (または “ファースト・スピーカー”) であるセッションの最小数をX定します。

**mode_detail.min_conwinners_target**
ローカル LU が競合敗者 (または “ビッダー”) であるセッションの最小数をX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h動化する前に、ローカル LU が待ちセッション要求を満たすかどうかをX定します (AP_NO または AP_YES)。

**mode_detail.drain_partner**
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h動化する前に、パートナー LU が待ちセッション要求を満たすかどうかをX定します (AP_NO または AP_YES)。

**mode_detail.auto_act**
パートナー LU とのセッション数変更 (CNOS) 交換の後で自動的に h 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H用するアクティブ競合勝者 (または “ファースト・スピーカー”) セッションの数。 (ローカル LU は、これらのいずれかのセッションをH用する前に送信権を要求する , 要はありません。)

**mode_detail.act_cl_count**
このモードをH用するアクティブ競合敗者 (または “ビッダー”) セッションの数。 (ローカル LU は、これらのいずれかのセッションをH用する前に送信権を要求しなければなりません。)

**mode_detail.sync_level**
モードによってサポートされる同期レベルを X 定します (AP_NONE、AP_CONFIRM、または AP_SYNCPT)。

**mode_detail.default_ru_size**
最大 RU サイズのデフォルト上限をH用するかどうかをX定します。このパラメーターに AP_YES の値を設定すると、 **define_mode** にX定された **mode_chars.max_ru_size_upp** フィールドは無 k され、最大 RU サイズの上限が、リンク BTU サイズから TH および RH のサイズを引いた値に設定されます。

AP_YES
AP_NO

**mode_detail.max_neg_sess_limit**
最大折衝可能セッション限度。ローカル LU が CNOS 処理中にターゲット LU としてH用できるモード名の最大セッション限度をX定します。

**mode_detail.max_rcv_ru_size**
最大受信 RU サイズ。

**mode_detail.pending_session_count**
保留セッション (セッション h 動化が完了するのを待っている) の数をX定します。

**mode_detail.termination_count**
直前の CNOS verb によってモード・セッション限度がゼロにリセットされた場合は、これらのセッションをH用していた会話、またはこれらのセッションのH用を待機していた会話が存在していた可能性があります。このフィールドは、このようなセッションがまだいくつ非 h 動化されていないかの数を示します。

**mode_detail.implicit**
暗黙定義 (AP_YES) または明示定義 (AP_NO) のどちらによって項目が書き込まれたかをX定します。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_MODE_NAME

AP_INVALID_PLU_NAME
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_MODE_DEFINITION

QUERY_MODE_DEFINITION は、先に DEFINE_MODE verb で渡された情報と SNA 定義のデフォルト・モードに関する情報の両方を戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のモードに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **mode_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10 ページの『ノードの照会』 を 2 照してください。

このリストは、**mode_name** 順に配列されます。配列はまず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (8 準の MIB 配列に準拠)。

AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

この verb は、定義情報のみを戻します。 QUERY_MODE verb は、パートナー LU をもつローカル LU がこのモードを H 用し O めるときに決定された情報を戻します。

## VCB 構造体

```
typedef struct query_mode_definition
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   mode_name[8];        /* mode name                     */
        unsigned char   description[RD_LEN]; /* resource description          */
} MODE_DEF_SUMMARY;

typedef struct mode_def_detail
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   mode_name[8];        /* mode name                     */
        MODE_CHARS      mode_chars;          /* mode characteristics          */
} MODE_DEF_DETAIL;

typedef struct mode_chars
{
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned short  max_ru_size_upp;     /* max RU size upper bound       */
        unsigned char   receive_pacing_win;  /* receive pacing window         */
        unsigned char   default_ru_size;     /* default RU size to maximize   */
```

```
                                        /* performance                    */
        unsigned short  max_neg_sess_lim;    /* max negotiable session limit   */
        unsigned short  plu_mode_session_limit;
                                        /* LU-mode session limit          */
        unsigned short  min_conwin_src;      /* min source contention winner   */
                                        /* sessions                       */
        unsigned char   cos_name[8];         /* class-of-service name          */
        unsigned char   cryptography;        /* cryptography                   */
        unsigned char   compression;         /* compression                    */
        unsigned short  auto_act;            /* initial auto-activation count  */
        unsigned short  min_conloser_src;    /* min source contention loser    */
        unsigned short  max_ru_size_low      /* maximum RU size lower bound    */
        unsigned short  max_receive_pacing_win;
                                        /* maximum receive pacing window  */
} MODE_CHARS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_MODE_DEFINITION

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

Xされた **mode_name** (以下のパラメーターを2照) は、戻された実際の情報の+O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無kされ、戻りリストはリスト内の最初の項目から+Oされます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**mode_name**
モード名。これは、セッション・グループのネットワーク特性をX定します。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**mode_def_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**mode_def_summary.mode_name**
8 バイトのモード名。これは、セッション・グループのネットワーク特性をX定します。

**mode_def_summary.description**
q 源の説明 (DEFINE_MODE でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**mode_def_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**mode_def_detail.mode_name**
モード名。これは、セッション・グループのネットワーク特性をX定します。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でO まる) で、右側に EBCDIC スペースが埋め込まれています。

## QUERY_MODE_DEFINITION

**mode_def_detail.mode_chars.description**
q 源の説明 (DEFINE_MODE でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**mode_def_detail.mode_chars.max_ru_size_upp**
このモード名をX定したセッションでH用される最大 RU サイズの上限。

**mode_def_detail.mode_chars.receive_pacing_win**
固定ペーシングをH用するときは、セッションのためのセッション・ペーシング・ウィンドウをX定します。適応ペーシングをH用するときは、優先最小ウィンドウ・サイズをX定します。

**mode_def_detail.mode_chars.default_ru_size**
最大 RU サイズのデフォルト上限をH用するかどうかをX定します。このパラメーターに AP_YES を設定すると、 **max_ru_size_upp** は無 k されます。

AP_YES
AP_NO

**mode_def_detail.mode_chars.max_neg_sess_lim**
最大折衝可能セッション限度。この値をH用して、ローカル LU とパートナー LU との間で、X定モード名について最大許容セッション数を折衝します。

**mode_def_detail.mode_chars.plu_mode_session_limit**
このモードについて最初に折衝するためのセッション限度。この値は優先セッション限度を示し、暗黙的な CNOS にH用されます。

範囲: 0 〜 32 767

**mode_def_detail.mode_chars.min_conwin_src**
このモードをH用するローカル LU によってh動化することができる競合勝者セッションの最小数。

範囲: 0 〜 32767

**mode_def_detail.mode_chars.cos_name**
サービス・クラス名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**mode_def_detail.mode_chars.cryptography**
このモードをH用するセッションで暗号化をH用するかどうかをX定します (AP_NONE または AP_MANDATORY)。

**mode_def_detail.mode_chars.compression**
このモードをH用するh動化済みセッションに圧縮をH用するかどうかをX定します。

**AP_COMP_PROHIBITED**
このモードのセッションでは、RLE 圧縮はサポートされません。

**AP_COMP_REQUESTED**
このモードのセッションでは、RLE 圧縮がサポートされ、要求されます (ただし , 須ではありません)。

**mode_def_detail.mode_chars.auto_act**
このモードで自動h動化されるセッションの番号を示します。この値は、暗黙的な CNOS にH用されます。

範囲: 0 〜 32767

**mode_def_detail.mode_chars.min_consloser_src**
このモードのいずれかのローカル LU によってh動化される競合敗者セッションの最小数をX定します。この値は、CNOS (セッション数の変更) 交換が暗黙的に+ Oされるときに H用されます。

範囲: 0 〜 32767

**mode_def_detail.mode_chars.max_ru_size_low**
このモードのセッションで送受信される RU の最大サイズの下限をX定します。この値は、セッションh動化~に最大 RU サイズを折衝するときに H用されます。

範囲: 0 〜 61140

**default_ru_size** を AP_YES に設定すると、このフィールドは無k されます。

**mode_def_detail.mode_chars.max_receive_pacing_win**
このモードのセッションの最大ペーシング・ウィンドウをX定します。適応ペーシングの場合、この値は、それに許される受信ペーシング・ウィンドウを制限するために H用されます。固定ペーシングの場合は、このフィールドは H用されません。隣接ノードは適応ペーシングをサポートしないということがX定されない限り、「プログラム」が常に適応ペーシングを H用することに注意してください。

範囲: 0 〜 32767

ゼロの値は、上限がないことを意味します。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_MODE_NAME

AP_INVALID_LIST_OPTION

ノードが+ Oされないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_MODE_TO_COS_MAPPING

QUERY_MODE_TO_COS_MAPPING は、モードから COS へのマッピングに関する情報を戻します。

この情報は定様式リストとして戻されます。特定のモードに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **mode_name** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**mode_name** 順に配列されます。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X 定された項目が存在するしないに関係なく)。

デフォルトの COS (認識されないモードがマップされる) が、DEFINE_MODE を H 用して上書きされると、 QUERY_MODE_TO_COS_MAPPING は、ヌルの **mode_name** (すべてゼロ) とデフォルトの COS も戻します。この項目は、配列の先頭に入れられます。

## VCB 構造体

```
typedef struct query_mode_to_cos_mapping
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this 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;
```

## 指定 Q i a ー ? ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_MODE_TO_COS_MAPPING

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
これは、リスト情報に何を戻すかを示します。つまり、X定された **mode_name** (以下のパラメーターを2照) は、戻された実際の情報の+O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無kされ、戻りリストはリスト内の最初の項目から+Oされます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から+Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から+Oされます。

**mode_name**
モード名。これは、セッション・グループのネットワーク特性をX定します。これは、8 バイト英数 z のタイプ A の EBCDIC 文z列 (文zでOまる) で、右側に EBCDIC スペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無kされます。このフィールドをすべてゼロに設定して、デフォルト COS のための項目であることを示すことができます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**mode_to_cos_mapping_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**mode_to_cos_mapping_data.mode_name**
8 バイトのモード名。これは、セッション・グループのネットワーク特性を X 定します。このフィールドをすべてゼロに設定した場合は、それがデフォルト COS のための項目であることを示します。

**mode_to_cos_mapping_data.cos_name**
モード名と関連するサービス・クラス名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_MODE_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_NMVT_APPLICATION

QUERY_NMVT_APPLICATION は、ネットワーク管理ベクトル移送 (NMVT) レベル・メッセージ用に登録されたアプリケーションのリストを戻します。この登録は、前もって REGISTER_NMVT_APPLICATION verb を発行することによって行われたものです (詳細については、 675ページの『第15章 管理サービス verb』 を 2 照)。

この情報はリストとして戻されます。特定のアプリケーションに関する情報または『固まり』 に分けられたリスト情報を入手するには、 **application** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

## VCB 構造体

```
typedef struct query_nmvt_application
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   secondary_rc;        /* secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   application[8];      /* application name               */
        unsigned short  ms_vector_key_type;  /* MS vector key accepted         */
                                             /* by appl                        */
        unsigned char   conversion_required; /* conversion to MDS_MU required  */
        unsigned char   reserv[5];           /* reserved                       */
        unsigned char   reserva[20];         /* reserved                       */
} NMVT_APPLICATION_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_NMVT_APPLICATION

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
これは、リスト情報に何を戻すかを示します。つまり、X定された **application** (以下のパラメーターを2照) は、戻された実際の情報の+O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無kされ、戻りリストはリスト内の最初の項目から+Oされます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から+Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から+Oされます。

**application**
アプリケーション名。この名前は、8 バイト英数 z のタイプ A の EBCDIC 文z 列、またはすべて EBCDIC ゼロです。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無kされます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**nmvt_application_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**nmvt_application_data.application**
登録されたアプリケーションの名前。この名前は、8 バイト英数 z のタイプ A の EBCDIC 文 z 列です。

**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た管理サービス・ベクトル・キー。アプリケーションを NMVT メッセージ用に登録するときは、そのアプリケーションがどの管理サービスベクトル・キーを受け入れるかを X 定します。NMVT アプリケーション登録の詳細については、 675ページの『第15章 管理サービス verb』 を 2 照してください。

**nmvt_application_data.conversion_required**
登録されたアプリケーションで、メッセージを NMVT から MDS_MU 形式に変換する , 要があるかどうかを X 定します (AP_YES または AP_NO)。アプリケーションを NMVT メッセージ用に登録するときは、この変換が , 要かどうかを X 定します。 NMVT アプリケーション登録の詳細については、 675ページの『第15章 管理サービス verb』 を 2 照してください。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_APPLICATION_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_NN_TOPOLOGY_NODE

この verb は Communications Server にのみ適用されます。

F

```
        unsigned char   format;             /* format                          */
        unsigned short  primary_rc;         /* primary return code             */
        unsigned long   secondary_rc;       /* secondary return code           */
        unsigned char   *buf_ptr;           /* pointer to buffer               */
        unsigned long   buf_size;           /* buffer size                     */
        unsigned long   total_buf_size;     /* total buffer size required      */
        unsigned short  num_entries;        /* number of entries               */
        unsigned short  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                       */
        unsigned long   frsn;               /* flow reduction sequence num     */
} QUERY_NN_TOPOLOGY_NODE;
```

**注: frsn** フィールドを非ゼロ値に設定すると、X定された FRSN より大きな FRSN を } つノード項目のみが戻されます。これをゼロに設定すると、すべてのノード項目が戻されます。

```
typedef struct nn_topology_node_summary
{
        unsigned short  overlay_size;       /* size of this 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
{
        unsigned short  overlay_size;       /* size of this entry              */
        unsigned char   node_name[17];      /* network qualified node name     */
        unsigned char   node_type;          /* node type                       */
        unsigned short  days_left;          /* days left until entry purged    */
        unsigned char   reserv1[2];         /* reserved                        */
        unsigned long   frsn;               /* flow reduction sequence num     */
        unsigned long   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;       /* node is branch aware            */
        unsigned char   reserva[20];        /* reserved                        */
} NN_TOPOLOGY_NODE_DETAIL;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_NN_TOPOLOGY_NODE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

**node_name**、**node_type**、および **frsn** を組み合わせたX定 (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**node_name**
ネットワーク・トポロジー・データベースから得られたネットワーク修飾ノード名。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**node_type**
ノードのタイプ。これは、以下のいずれかの値にすることができます。

AP_NETWORK_NODE
AP_VRN

**node_type** がT明の場合は、AP_LEARN_NODE をX定する , 要があります。

**frsn** フロー縮小順序番号。これが非ゼロであれば、この値以上の FRSN をもつノードのみが戻されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**nn_topology_node_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**nn_topology_node_summary.node_name**
ネットワーク・トポロジー・データベースから得られたネットワーク修飾ノード名。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**nn_topology_node_summary.node_type**
ノードのタイプ。以下のいずれかの値に設定されます。

AP_NETWORK_NODE
AP_VRN

**nn_topology_node_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**nn_topology_node_detail.node_name**
ネットワーク・トポロジー・データベースから得られたネットワーク修飾ノード名。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**nn_topology_node_detail.node_type**
ノードのタイプ。以下のいずれかの値に設定されます。

AP_NETWORK_NODE
AP_VRN

**nn_topology_node_detail.days_left**
このノード項目をトポロジー・データベースから削除するまでの日数。ローカル・ノード項目の場合、このフィールドはゼロに設定されます (この項目が削除されることはありません)。

**nn_topology_node_detail.frsn**
フロー縮小順序番号。この番号は、この q 源がローカル・ノードで最後に更新されたことを示します。

## QUERY_NN_TOPOLOGY_NODE

**nn_topology_node_detail.rsn**
q 源順序番号。この番号は、この q 源を所有するネットワーク・ノードによってd り当てられます。

**nn_topology_node_detail.rar**
ノードの経路追加レジスタンス。

**nn_topology_node_detail.status**
ノードの状況をX定します。この値は、AP_UNCONGESTED にするか、または以下の 1 つの値またはいくつかの値を OR で結合した値にすることができます。

**AP_CONGESTED**
ISR セッションの数が、 **isr_sessions_upper_threshold** よりも大きくなっています。

**AP_ERR_DEPLETED**
エンドポイント・セッションの数が、最大X定数に達しました。

**AP_IRR_DEPLETED**
ISR セッションの数が最大数に達しました。

**AP_QUIESCING**
STOP_NODE、またはタイプ AP_QUIESCE か AP_QUIESCE_ISR が発行されました。

**nn_topology_node_detail.function_support**
どの機能がサポートされるかを示します。これは、以下の 1 つまたは複数の値になります。

**AP_PERIPHERAL BORDER_NODE**
周辺ボーダー・ノード機能がサポートされます。

**AP_EXTENDED BORDER_NODE**
H張ボーダー・ノード機能がサポートされます。

**AP_CDS**
ノードは、中央ディレクトリー・サーバー機能をサポートします。

**AP_GATEWAY**
ノードは、ゲートウェイ・ノードです。(この機能は、まだアーキテクチャー的には定義されていません。)

**AP_INTERCHANGE_NODE**
このノードは、ゲートウェイ・ノードです。(この機能は、まだアーキテクチャー的には定義されていません。)

**AP_ISR**
ノードは、中間セッション経路X定をサポートします。

**AP_HPR**
ノードは、高性能経路X定の基本機能をサポートします。

**AP_RTP_TOWER**
ノード-は、HPR の RTP タワーをサポートします。

**AP_CONTROL_OVER_RTP_TOWER**
ノードは、RTP タワーを介した制御フローをサポートします。

**注:** AP_CONTROL_OVER_RTP_TOWER は、 AP_HPR と AP_RTP_TOWER の両方の設定に対応します。

**nn_topology_node_detail.branch_aware**
ノードがブランチ・アウェアであるかどうかをX定します。

**AP_NO**
ノードはブランチ・アウェアでありません。

**AP_YES**
ノードはブランチ・アウェアです。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_NODE

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_NN_TOPOLOGY_STATS

この verb は Communications Server にのみ適用されます。

QUERY_NN_TOPOLOGY_STATS は、トポロジー・データベースに関する統計情報を戻し、ネットワーク・ノードでのみ発行されます。

## VCB 構造体

```
typedef struct query_nn_topology_stats
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned long   max_nodes;           /* max num of nodes in database  */
        unsigned long   cur_num_nodes;       /* current number of nodes in    */
                                             /* database                      */
        unsigned long   node_in_tdus;        /* number of TDUs received       */
        unsigned long   node_out_tdus;       /* number of TDUs sent           */
        unsigned long   node_low_rsns;       /* node updates received with    */
                                             /* low RSNs                      */
        unsigned long   node_equal_rsns;     /* node updates in with          */
                                             /* equal RSNs                    */
        unsigned long   node_good_high_rsns; /* node updates in with          */
                                             /* high RSNs                     */
        unsigned long   node_bad_high_rsns;  /* node updates in with          */
                                             /* high and odd RSNs             */
        unsigned long   node_state_updates;  /* number of node updates sent   */
        unsigned long   node_errors;         /* number of node entry          */
                                             /* errors found                  */
        unsigned long   node_timer_updates;  /* number of node records built  */
                                             /* due to timer updates          */
        unsigned long   node_purges;         /* num node records purged       */
        unsigned long   tg_low_rsns;         /* TG updates received with      */
                                             /* low RSNs                      */
        unsigned long   tg_equal_rsns;       /* TG updates in with equal RSNs */
        unsigned long   tg_good_high_rsns;   /* TG updates in with high RSNs  */
        unsigned long   tg_bad_high_rsns;    /* TG updates in with high       */
                                             /* and odd RSNs                  */
        unsigned long   tg_state_updates;    /* number of TG updates sent     */
        unsigned long   tg_errors;           /* number of TG entry errors     */
                                             /* found                         */
        unsigned long   tg_timer_updates;    /* number of node records        */
                                             /* built due to timer updates    */
        unsigned long   tg_purges;           /* num node records purged       */
        unsigned long   total_route_calcs;   /* num routes calculated for COS */
        unsigned long   total_route_rejs;    /* num failed route calculations */
        unsigned long   total_tree_cache_hits;
                                             /* total num of tree cache hits  */
        unsigned long   total_tree_cache_misses;
                                             /* total num of tree cache       */
                                             /* misses                        */
        unsigned counter  total_tdu_wars;    /* total number TDU war          */
        unsigned char   reserva[16];         /* reserved                      */
} QUERY_NN_TOPOLOGY_STATS;
```

## 指定 Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_NN_TOPOLOGY_STATS

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_OK

**max_nodes**

トポロジー・データベース内のノード・レコードの最大数 (ゼロは限度がないことを意味します)。

**cur_num_nodes**

このノードのトポロジー・データベースに入っている現在のノードの数。この値がノードの許容最大数を超える場合には、アラートが発行されます。

**node_in_tdus**

このノードによって受信されたトポロジー・データベース更新 (TDU) の合計数。

**node_out_tdus**

最後の初期設定以降にこのノードによって作成されたトポロジー・データベース更新 (TDU) のうち、すべての隣接ネットワーク・ノードに送信されるものの合計数。

**node_low_rsns**

このノードによって受信されたトポロジー・ノード更新のうち、現行 RSN よりも小さい RSN もつものの合計数。偶数の RSN と奇数の RSN の両方がこの数に含まれます。(これらの TDU はエラーではなく、 TDU がすべての隣接ネットワーク・ノードに同報通信された場合に生じます。このノードのトポロジー・データベースの更新は行われず、このノードは、この小さい RSN を送信した隣接ノードに、それより大きい RSN を } つ TDU を送信します。)

**node_equal_rsns**

このノードによって受信されたトポロジー・ノード更新のうち、現行 RSN と等しい RSN をもつものの合計数。偶数の RSN と奇数の RSN の両方がこのカウントに含まれます。(これらの TDU はエラーではなく、 TDU がすべての隣接ネットワーク・ノードに同報通信された場合に生じます。このノードのトポロジー・データベースの更新は行われません。)

**node_good_high_rsns**

このノードによって受信されたトポロジー・ノード更新のうち、現行 RSN よりも大きい RSN をもつものの合計数。ノードはそのトポロジーを更新し、TDU をすべての隣接ネットワーク・ノードに同報通信します。このノードはすでに更新結果をもっているため、この更新結果の送信側に TDU を送信する，要はありません。

**node_bad_high_rsns**

このノードによって受信されたトポロジー・ノード更新のうち、現行 RSN よ

りも大きい奇数の RSN をもつものの合計数。これらの更新は、 APPN ネットワーク・ノードの 1 つによって検出されたトポロジー矛盾を示しています。ノードはそのトポロジーを更新し、 TDU をすべての隣接ネットワーク・ノードに同報通信します。

**node_state_updates**

内 t で検出されたノード状態変更の結果作成されたトポロジー・ノード更新のうち、 APPN トポロジーと経路X定に影響を与えるものの合計数。更新結果は7^9　　。ド　　ドードにに.　　。　ノ　ノ　ノ　ノのの

も

ーク・ノードの 1 つによって検出されたトポロジー矛盾を示しています。ノードはそのトポロジーを更新し、 TDU をすべての隣接ネットワーク・ノードに同報通信します。

**tg_state_updates**
内 t で検出されたノード状態変更の結果作成されたトポロジー・ノード更新のうち、 APPN トポロジーと経路 X 定に影響を与えるものの合計数。更新結果は、 TDU によってすべての隣接ネットワーク・ノードに送信されます。

**tg_errors**
このノードによって検出されたトポロジー TG ノード更新矛盾の合計数。これは、このノードがそのトポロジー・データベースを更新しようとして、データ矛盾を検出したときに起こります。このノードは、現行 RSN を次の奇数番号に増分して TDU を作成し、それをすべての隣接ネットワーク・ノードに同報通信します。

**tg_timer_updates**
タイマー更新のため、このノードの q 源について作成されたトポロジー TG 更新の合計数。更新結果は、 TDU によってすべての隣接ネットワーク・ノードに送信されます。これらの更新によって、他のネットワーク・ノードがこのノードの q 源をトポロジー・データベースから削除しないことが N 認されます。

**tg_purges**
このノードのトポロジー・データベースから除去されたトポロジー TG レコードの合計数。これは、ノード・レコードが X 定 ~ 間内に更新されなかったときに起こります。所有側のノードは、ネットワーク・トポロジー内に D しておきたい q 源に対する更新を同報通信しなければなりません。

**total_route_calcs**
最後の初期設定以降、すべてのサービス・クラスについて計；された経路の数。

**total_route_rejs**
最後の初期設定以降、すべてのサービス・クラスに対する経路要求のうち、計；できなかったものの数。

**total_tree_cache_hits**
キャッシュ済みの経路 X 定ツリーによって満足された経路計；の数。それぞれの経路がいくつかのツリーの検査を，要とすることがあるため、この数が；出経路の合計数より大きくなる可能性があることに注意してください。

**total_tree_cache_misses**
キャッシュ済みの経路 X 定ツリーによって満足されなかったために、新しい経路 X 定ツリーの作成が，要になった経路計；の数。

**total_tdu_wars**
ローカル・ノードによって検出されて防 _ された TDU 競合の数。

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

## QUERY_NN_TOPOLOGY_STATS

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_NN_TOPOLOGY_TG

この verb は Communications Server にのみ適用されます。

F ネットワーク・ノードは、ネットワークのネットワーク・ノードに関する情報や、 VRN およびネットワーク・ノード間 TG に関する情報を保管するネットワーク・トポロジー・データベースを維 } しています。 QUERY_NN_TOPOLOGY_TG は、このデータベース内の TG 項目に関する情報を戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のノードに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **owner**、**owner_type**、 **dest**、**dest_type**、 **tg_num**、および **frsn** フィールドを設定する , 要があります。それ以 0 の場合 (**list_options** フィールドが AP_FIRST_IN_LIST に設定されていれば)、これらのフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**owner**、 **owner_type**、**dest**、 **dest_type**、**tg_num**、および **frsn** 順に配列されています。**owner** 名および **dest** 名は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。 **owner_type** と **dest_type** は、AP_NETWORK_NODE、AP_VRN 順になります。**tg_num** と **frsn** は、数値順になります。

AP_LIST_INCLUSIVE を選択すると、戻りリストはその名前の最初の有効なレコードから + O されます。

AP_LIST_FROM_NEXT を選択すると、リストは、 X 定された名前の次の名前をもつ最初の有効なレコードから + O されます。

**frsn** フィールド (フロー縮小順序番号) を非ゼロ値に設定すると、この値より高い FRSN 値を } つデータベース項目のみが戻されます。このため、まずノードの現行 FRSN 値を入手することによって、一貫性のあるトポロジー・データベースをいくつかの “固まり” に分けて戻すことができます。 これは以下のように行われます。

1. QUERY_NODE を発行します。これは、ノードの現行 FRSN を戻します。
2. , 要な数だけ QUERY_NN_TOPOLOGY_TG (FRSN をゼロに設定) を発行して、すべてのデータベース項目をいくつかの “固まり” に分けて入手します。
3. 再度 QUERY_NODE を発行し、新規の FRSN とステップ 1 で戻された FRSN とを比 S します。
4. この 2 つの FRSN が異なっている場合は、データベースが変更されています。このため、FRSN を、ステップ 1 で提供された FRSN より 1 だけ大きく設定して、QUERY_NN_TOPOLOGY_TG を発行します。

## VCB 構造体

```
typedef struct query_nn_topology_tg
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
```

```
        unsigned char   format;             /* format                          */
        unsigned short  primary_rc;         /* primary return code             */
        unsigned long   secondary_rc;       /* secondary return code           */
        unsigned char   *buf_ptr;           /* pointer to buffer               */
        unsigned long   buf_size;           /* buffer size                     */
        unsigned long   total_buf_size;     /* total buffer size required      */
        unsigned short  num_entries;        /* number of entries               */
        unsigned short  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                        */
        unsigned long   frsn;               /* flow reduction sequence num     */
} QUERY_NN_TOPOLOGY_TG;

typedef struct topology_tg_summary
{
        unsigned short  overlay_size;       /* size of this 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                        */
        unsigned long   frsn;               /* flow reduction sequence num     */
} TOPOLOGY_TG_SUMMARY;

typedef struct topology_tg_detail
{
        unsigned short  overlay_size;       /* size of this 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                        */
        unsigned long   frsn;               /* flow reduction sequence num     */
        unsigned short  days_left;          /* days left until entry purged    */
        LINK_ADDRESS    dlc_data            /* DLC signalling data             */
        unsigned long   rsn;                /* resource sequence number        */
        unsigned char   status;             /* node status                     */
        TG_DEFINED_CHARS tg_chars;          /* TG characteristics              */
        unsigned char   subarea_number[4];  /* subarea number                  */
        unsigned char   tg_type;            /* TG type                         */
        unsigned char   intersubnet_tg;     /* intersubnet TG?                 */
        unsigned char   cp_cp_session_active;
                                            /* CP-CP session is active         */
        unsigned char   branch_tg;          /* TG is a branch TG               */
        unsigned char   reserva[12];        /* reserved                        */
} TOPOLOGY_TG_DETAIL;

typedef struct link_address
{
        unsigned short  length;             /* length                          */
        unsigned short  reserve1;           /* reserved                        */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                            /* address                         */
} LINK_ADDRESS;
```

**注: frsn** フィールドを非ゼロ値に設定すると、その FRSN をもつノード項目のみが戻されます。これをゼロに設定すると、すべてのノード項目が戻されます。

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**
AP_QUERY_NN_TOPOLOGY_TG

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

**owner**、**owner_type**、**dest**、 **dest_type**、**tg_num**、および **frsn** を組み合わせたX定 (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**owner** TG の発信側ノードの名前。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**owner_type**
TG を所有するノードのタイプ。これは、以下のいずれかの値にすることができます。

AP_NETWORK_NODE
AP_VRN

**owner_type** がT明の場合は、AP_LEARN_NODE をX定する，要があります。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**dest** TG の完全修飾宛先ノード名。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**dest_type**
この TG の宛先ノードのタイプ。これは、以下のいずれかの値にすることができます。

AP_NETWORK_NODE
AP_VRN

**dest_type** がT明の場合は、 AP_LEARN_NODE をX定する，要があります。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**tg_num**
TG と関連する番号。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**frsn** フロー縮小順序番号。これが非ゼロであれば、この値以上の FRSN をもつノードのみが戻されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**topology_tg_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**topology_tg_summary.owner**
TG の発信側ノードの名前。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**topology_tg_summary.owner_type**
TG を所有するノードのタイプ。以下のいずれかの値に設定されます。

AP_NETWORK_NODE
AP_VRN

**topology_tg_summary.dest**
TG の完全修飾宛先ノード名。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**topology_tg_summary.dest_type**
この TG の宛先ノードのタイプ。以下のいずれかの値に設定されます。

AP_NETWORK_NODE
AP_VRN

**topology_tg_summary.tg_num**
TG と関連する番号。

**topology_tg_summary.frsn**
フロー縮小順序番号。この番号は、この q 源がローカル・ノードで最後に更新されたことを示します。

**topology_tg_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**topology_tg_detail.owner**
TG の発信元ノードの名前。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**topology_tg_detail.owner_type**
TG を所有するノードのタイプ。以下のいずれかの値に設定されます。

AP_NETWORK_NODE
AP_VRN

**topology_tg_detail.dest**
TG の完全修飾宛先ノード名。この名前の長さは 17 バイトで、1 つのドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**topology_tg_detail.dest_type**
この TG の宛先ノードのタイプ。以下のいずれかの値に設定されます。

AP_NETWORK_NODE
AP_VRN

**topology_tg_detail.tg_num**
TG と関連する番号。

**topology_tg_detail.frsn**
フロー縮小順序番号。この番号は、この q 源がローカル・ノードで最後に更新されたことを示します。

**topology_node_detail.days_left**
このノード項目をトポロジー・データベースから削除するまでの日数。

**topology_tg_detail.dlc_data.length**
VRN への接続の DLC アドレスの長さ (**dest_type** が AP_VRN でなければ、ゼロに設定されます)。

**topology_tg_detail.dlc_data.address**
VRN への接続の DLC アドレス。 **dest_type** が AP_VRN でない場合は、ゼロに設定されます。

**topology_tg_detail.rsn**
q 源順序番号。この番号は、この q 源を所有するネットワーク・ノードによって d り当てられます。

**topology_tg_detail.status**
TG の状況を X 定します。これは、以下のいずれかの値、または複数の値を OR 結合した値になります。

AP_TG_OPERATIVE
AP_TG_QUIESCING
AP_TG_GARBAGE_COLLECT
AP_TG_CP_CP_SESSIONS
AP_TG_HPR
AP_TG_RTP
AP_TG_NONE

**topology_tg_detail.tg_chars**
TG 特性。

**topology_tg_detail.subarea_number**
TG の所有者または宛先ノードがサブエリア可能であれば、このフィールドには、サブエリア可能ノードの TG と関連するリンク・ステーションを所有するタイプ 4 またはタイプ 5 のサブエリア番号が含まれています。それ以 0 の場合は、このフィールドはすべて 2 進ゼロに設定されます。

**topology_tg_detail.tg_type**
TG のタイプ。このフィールドは、以下のいずれかの値になります。

AP_APPN_OR_BOUNDARY_TG
APPN TG または境＆機能ベースの TG

AP_INTERCHANGE_TG
交換 TG

AP_VIRTUAL_ROUTE_BASED_TG
仮想経路ベースの TG

AP_UNKNOWN
トポロジーに報告されたこの TG の TG タイプは T 明です。

**topology_tg_detail.intersubnet.tg**
この TG はサブネットワーク間 TG ですか?

AP_YES
AP_NO

**topology_tg_detail.cp_cp_session_active**
所有側ノードの競合勝者 CP-CP セッションがアクティブであるかどうかを X 定します (AP_UNKNOWN、AP_NO、または AP_YES)。

**branch_tg**
この TG がブランチ TG であるかどうかを X 定します。

**AP_NO**
この TG はブランチ TG ではありません。

**AP_YES**
この TG はブランチ TG です。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TG

AP_INVALID_ORIGIN_NODE
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_NODE

QUERY_NODE は、ノード固有の情報と統計を戻します。 QUERY_NODE は、実行 ~ に動的に決定される情報を戻すだけではなく、ノードの初期設定 ~ に設定されるパラメーターも戻します。

## VCB 構造体

### Format 2

```
typedef struct query_node
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        CP_CREATE_PARMS cp_create_parms;     /* create parameters             */
        unsigned long   up_time;             /* time since node started       */
        unsigned long   mem_size;            /* size of memory available      */
        unsigned long   mem_used;            /* size of memory used           */
        unsigned long   mem_warning_threshold;
                                             /* memory constrained            */
                                             /* threshold                     */
        unsigned long   mem_critical_threshold;
                                             /* memory critical threshold     */
        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.  One or more of  */
        unsigned long   nn_frsn;             /* NN flow reduction             */
                                             /* sequence number               */
        unsigned long   nn_rsn;              /* Resource sequence number      */
        unsigned short  def_ls_good_xids;    /* Good XIDs for defined         */
                                             /* link stations                 */
        unsigned short  def_ls_bad_xids;     /* Bad XIDs for defined          */
                                             /* link stations                 */
        unsigned short  dyn_ls_good_xids;    /* Good XIDs for dynamic         */
                                             /* link stations                 */
        unsigned short  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 support for registration  */
                                             /* of DLUS-served LUs reserved   */
        unsigned char   reserva[19];         /* reserved                      */
        unsigned char   fq_nn_server_name[17];
                                             /* FQ name of NN server          */
        unsigned long   current_isr_sessions;
                                             /* current ISR sessions          */
        unsigned char   nn_functions2;       /* NN functions continued        */
        unsigned char   branch_ntwk_arch_version;
                                             /* branch network architecture   */
                                             /* version supported             */
        unsigned char   reservb[28];         /* reserved                      */
} QUERY_NODE;

typedef struct cp_create_parms
{
        unsigned short  crt_parms_len;       /* length of CP_CREATE_PARMS     */
        unsigned char   description[RD_LEN]; /* resource description          */
        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                       */
        unsigned short  max_locates;         /* max locates node can process  */
        unsigned short  dir_cache_size;      /* directory cache size          */
                                             /* (reserved) if not NN)         */
        unsigned short  max_dir_entries;     /* max directory entries         */
        unsigned short  locate_timeout;      /* locate timeout in seconds     */
        unsigned char   reg_with_nn;         /* register resources with NNS   */
        unsigned char   reg_with_cds;        /* resource registration with    */
                                             /* CDS                           */
        unsigned short  mds_send_alert_q_size;
                                             /* size of MDS send alert queue  */
        unsigned short  cos_cache_size;      /* number of COS definitions     */
        unsigned short  tree_cache_size;     /* Topology Database routing     */
                                             /* tree cache size               */
        unsigned short  tree_cache_use_limit;
                                             /* num times tree can be used    */
        unsigned short  max_tdm_nodes;       /* max num nodes that can be     */
                                             /* stored in Topology Database   */
        unsigned short  max_tdm_tgs;         /* max num TGs that can be       */
                                             /* stored in Topology Database   */
        unsigned long   max_isr_sessions;    /* max ISR sessions              */
        unsigned long   isr_sessions_upper_threshold;
                                             /* upper threshold for ISR sess  */
        unsigned long   isr_sessions_lower_threshold;
                                             /* lower threshold for ISR sess  */
        unsigned short  isr_max_ru_size;     /* max RU size for ISR           */
        unsigned short  isr_rcv_pac_window;  /* ISR rcv 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;              /* HPR path-switch controller?   */
        unsigned char   discovery_support;   /* Discovery function utilized   */
        unsigned char   discovery_group_name[8];
                                             /* Group name for Discovery      */
        unsigned char   implicit_lu_0_to_3;  /* Implicit LU 0 to 3 support    */
        unsigned char   default_preference;  /* Default routing preference    */
        unsigned char   anynet_supported;    /* level of AnyNet support       */
        unsigned short  max_ls_exception_events;
                                             /* maximum LS Exception events   */
        unsigned char   comp_in_series;      /* compression in series allowed */
        unsigned char   max_compress_lvl;    /* maximum compression level     */
        unsigned char   node_spec_data_len;  /* length of node specific data  */
        unsigned char   ptf[64];             /* program temporary fix array   */
} CP_CREATE_PARMS;
```

### Format 1 (バック・レベル)

```
typedef struct query_node
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        CP_CREATE_PARMS cp_create_parms;     /* create parameters             */
        unsigned long   up_time;             /* time since node started       */
        unsigned long   mem_size;            /* size of memory available      */
```

```
        unsigned long   mem_used;            /* size of memory used           */
        unsigned long   mem_warning_threshold;
                                             /* memory constrained            */
                                             /* threshold                     */
        unsigned long   mem_critical_threshold;
                                             /* memory critical threshold     */
        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.  One or more of  */
        unsigned long   nn_frsn;             /* NN flow reduction             */
                                             /* sequence number               */
        unsigned long   nn_rsn;              /* Resource sequence number      */
        unsigned short  def_ls_good_xids;    /* Good XIDs for defined         */
                                             /* link stations                 */
        unsigned short  def_ls_bad_xids;     /* Bad XIDs for defined          */
                                             /* link stations                 */
        unsigned short  dyn_ls_good_xids;    /* Good XIDs for dynamic         */
                                             /* link stations                 */
        unsigned short  dyn_ls_bad_xids;     /* Bad XIDs for dynamic          */
                                             /* link stations                 */
        unsigned char   dlur_release_level;  /* Current DLUR release level    */
        unsigned char   reserva[19];         /* reserved                      */
} QUERY_NODE;
```

### Format 0 (バック・レベル)

```
typedef struct query_node
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        CP_CREATE_PARMS cp_create_parms;     /* create parameters             */
        unsigned long   up_time;             /* time since node started       */
        unsigned long   mem_size;            /* size of memory available      */
        unsigned long   mem_used;            /* size of memory used           */
        unsigned long   mem_warning_threshold;
                                             /* memory constrained            */
                                             /* threshold                     */
        unsigned long   mem_critical_threshold;
                                             /* memory critical threshold     */
        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.  One or more of  */
        unsigned long   nn_frsn;             /* NN flow reduction             */
                                             /* sequence number               */
        unsigned long   nn_rsn;              /* Resource sequence number      */
        unsigned short  def_ls_good_xids;    /* Good XIDs for defined         */
                                             /* link stations                 */
        unsigned short  def_ls_bad_xids;     /* Bad XIDs for defined          */
                                             /* link stations                 */
        unsigned short  dyn_ls_good_xids;    /* Good XIDs for dynamic         */
                                             /* link stations                 */
        unsigned short  dyn_ls_bad_xids;     /* Bad XIDs for dynamic          */
                                             /* link stations                 */
        unsigned char   dlur_release_level;  /* Current DLUR release level    */
        unsigned char   reserva[19];         /* reserved                      */
} QUERY_NODE;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**
AP_QUERY_NODE

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

このフィールドをゼロに設定すると、 **def_ls_good_xids**、**def_ls_bad_xids**、**dyn_ls_good_xids**、および **dyn_ls_bad_xids** の 4 つのフィールドが、Unsigned_COUNTER ではなく、無 d 号短形式になります。

このフィールドを 2 に設定すると、**fq_nn_server_name** と **current_isr_sessions** のフィールドが、記述されているとおりにH用されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**cp_create_parms.crt_parms_len**
パラメーター作成構造の長さ。

**cp_create_parms.description**
q 源の説明。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**cp_create_parms.node_type**
これは、常に以下の値になります。

AP_END_NODE
AP_NETWORK_NODE
AP_LEN_NODE
AP_BRANCH_NETWORK_NODE

**cp_create_parms.fqcp_name**
ノードの 17 バイトの完全修飾制御点名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**cp_create_parms.cp_alias**
ローカルH用制御点別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**cp_create_parms.mode_to_cos_map_supp**
モードから COS へのマッピングがノードによってサポートされているかどうかをX定します (AP_YES または AP_NO)。これを AP_YES に設定した場合

は、DEFINE_MODE verb でX定された COS は、 SNA 定義 COS であるか、または DEFINE_COS verb を発行して定義されていなければなりません。

**cp_create_parms.mds_supported**
管理サービスが複数ドメイン・サポートと管理サービス機能をサポートするかどうかをX定します (AP_YES または AP_NO)。

**cp_create_parms.node_id**
XID 交換でH用されたノード ID。これは、4 バイトの16 進数文 z 列です。

**cp_create_parms.max_locates**
ノードが処理できるロケートの最大数。

**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。

**cp_create_parms.max_dir_entries**
ディレクトリー項目の最大数。このフィールドがゼロに設定されている場合は、限度がありません。

**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~間をX定します (C単位)。ゼロの値は、検索がタイムアウトにならないことを示します。

**cp_create_parms.reg_with_nn**
q 源をネットワーク・ノード・サーバーに登録するかどうかをX定します (AP_YES または AP_NO)。登録に失敗しても、ノード初期設定の正常終了には影響を与えません。詳細については、
601ページの『REGISTRATION_FAILURE』 を2照してください。このフィールドは、EN と BrNN では、異なって解釈されます。

エンド・ノード:

**AP_NO**
ノードは、LU を自分の NN サーバーに登録しません。NNS は、すべての同報通信検索をエンド・ノードに転送します。

**AP_YES**
ノードは、すべてのローカル従属 LU (NNS がオプション・セット 1116 をサポートしている場合) とすべてのローカル独立 LU を自分の NNS に登録します。 NNS は、宛先X定されたロケートのみをそれに転送します (登録できなかった従属 LU を所有している場合を除く)。

ブランチ・ネットワーク・ノード:

**AP_REGISTER_NONE**
ノードは、LU を自分の NN サーバーに登録しません。

**AP_REGISTER_ALL**
ノードは、すべてのローカル従属 LU (それが DLUR フル・マルチ・サブネットをサポートし、 NNS がオプション・セット 1116 をサポートしている場合) とすべてのドメイン独立 LU を自分の NNS に登録します。

**AP_REGISTER_LOCAL_ONLY**
ノードは、すべてのローカル従属 LU (それが DLUR フル・マルチ・

サブネットをサポートし、 NNS がオプション・セット 1116 をサポートしている場合) とすべてのローカル独立 LU を自分の NNS に登録します。

**cp_create_parms.reg_with_cds**
q 源を中央ディレクトリー・サーバー (CDS) に登録できるかどうかを X 定します。このフィールドは、EN、NN、または BrNN では、異なって解釈されます。

エンド・ノード: NNS を CDS エンド・ノード q 源に登録できるかどうかを X 定します。 **reg_with_nn** を AP_NONE に設定すると、このフィールドは無 k されます。

**AP_NO**
EN q 源を CDS に登録することはできません。

**AP_YES**
EN q 源を CDS に登録することができます。

ネットワーク・ノード: ローカル q 源とドメイン q 源 (所有側の EN によって CDS への登録が許可されている) を CDS に登録できるかどうかを X 定します。

**AP_NO**
ローカル q 源またはドメイン q 源を CDS に登録することはできません。

**AP_YES**
ローカル q 源またはドメイン q 源を CDS に登録することができます。登録に失敗しても、START_NODE verb の正常終了には影響を与えません。

ブランチ・ネットワーク・ノード: NNS を CDS BrNN q 源 (BrNN に対してローカル、または BrNN のドメインからローカルになっている) に登録できるかどうかを X 定します。 **reg_with_nn** を AP_NO に設定すると、このフィールドは無 k されます。

**AP_REGISTER_NONE**
ノードは、LU を自分の NN サーバーに登録しません。

**AP_REGISTER_ALL**
ノードは、すべてのローカル従属 LU (それが DLUR フル・マルチ・サブネットをサポートし、 NNS がオプション・セット 1116 をサポートしている場合) とすべてのドメイン独立 LU を自分の NNS に登録します。

**AP_REGISTER_LOCAL_ONLY**
ノードは、すべてのローカル従属 LU (それが DLUR フル・マルチ・サブネットをサポートし、 NNS がオプション・セット 1116 をサポートしている場合) とすべてのローカル独立 LU を自分の NNS に登録します。

**cp_create_parms.mds_send_alert_q_size**
MDS 送信アラート待ち行列のサイズ。この限度に達すると、MDS 構成要素は、待ち行列で最も古い項目を削除します。

**cp_create_parms.cos_cache_size**
COS データベース重みキャッシュのサイズ。

**cp_create_parms.tree_cache_size**
トポロジー・データベース経路X定ツリー・キャッシュのサイズ。

**cp_create_parms.tree_cache_use_limit**
キャッシュ済みツリーの最大H用回数。この数を超えると、ツリーは破棄され、再計；されます。このため、ノードは、重みが等しい経路間でセッションを均衡化することができます。値が小さければ、ロード・バランシングはうまくいきますが、 h 動化待ち ~ 間が長くなります。

**cp_create_parms.max_tdm_nodes**
トポロジー・データベースに J 納できるノードの最大数 (ゼロは限度がないことを意味します)。

**cp_create_parms.max_tdm_tgs**
トポロジー・データベースに J 納できる TG の最大数 (ゼロは限度がないことを意味します)。

**cp_create_parms.max_isr_sessions**
ノードがただちに 2 加できる ISR セッションの最大数。

**cp_create_parms.isr_sessions_upper_threshold**
**cp_create_parms.isr_sessions_lower_threshold** を 2 照してください。

**cp_create_parms.isr_sessions_lower_threshold**
上限および下限のしきい値がノードの輻輳状況を制御します。 ISR セッション数が上限のしきい値を超えると、ノード状態は非輻湊から輻湊に変わります。ISR セッション数が下限のしきい値より小さくなると、ノード状態は非輻湊に戻ります。

**cp_create_parms.isr_max_ru_size**
中間セッションのためにサポートされている最大 RU サイズ。

**cp_create_parms.isr_rcv_pac_window**
中間セッションのための推奨受信ペーシング・ウィンドウ・サイズ。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H用されます。

**cp_create_parms.store_endpt_rscvs**
RSCV を診断の目的で J 納するかどうかをX定します (AP_YES または AP_NO)。

**cp_create_parms.store_isr_rscvs**
RSCV を診断の目的で J 納するかどうかをX定します (AP_YES または AP_NO)。

**cp_create_parms.store_dlur_rscvs**
ノードが診断の目的で RSCV を J 納するかどうかをX定します (AP_YES または AP_NO)。このフィールドを AP_YES に設定すると、RSCV は、QUERY_DLUR_LU verb で戻されます。

**cp_create_parms.dlur_support**
ノードによって提供される DLUR サポートのレベルをX定します。これはビット・フィールドであり、以下の値にすることができます。

**AP_NO**
DLUR はサポートされません。

**AP_YES**
DLUR フル・マルチ・サブネットがサポートされます。

**(AP_YES | AP_LIMITED_DLUR_MULTI_SUBNET)**
DLUR 限定マルチ・サブネットがサポートされます。これは、ノードがエンド・ノードである場合にのみ有効です。

**cp_create_parms.pu_conc_support**
PU 集信がサポートされるかどうかを示します (常に AP_NO)。

**cp_create_parms.nn_rar**
ネットワーク・ノードの経路追加レジスタンス。

**cp_create_parms.hpr_support**
ノードによって提供される HPR サポートのレベルを X 定します (AP_NONE、AP_BASE、または AP_RTP)。

**cp_create_parms.mobile**
ノードが HPR パス・スイッチ・コントローラーであるかどうかをX定します (AP_YES または AP_NO)。 **cp_create_parms.hpr_support** フィールドを AP_RTP に設定しないと、このフィールドは予約済みになります。

**cp_create_parms.discovery_support**
ディスカバリー機能がこのノードによってH用されるかどうかをX定します。

**AP_DISCOVERY_CLIENT**
ディスカバリー・クライアント機能がこのノードによってH用されます。

**AP_DISCOVERY_SERVER**
ディスカバリー・サーバー機能がこのノードによってH用されます。

**cp_create_parms.discovery_group_name**
ノードがH用したディスカバリー機能でのグループ名を示します。このフィールドをすべてゼロに設定すると、デフォルトのグループ名がH用されます。

**cp_create_parms.implicit_lu_0_to_3**
ノードが ACTLU 別タイプ 0 ～ 3 の LU の暗黙定義をサポートするかどうかをX定します (AP_YES または AP_NO)。

**cp_create_parms.default_preference**
このノードからセッションを + Oする際の経路X定優先メソッドをX定します。

**注:** これは、DEFINE_PARTNER_LU verb によって LU 単位で上書きできます。

このフィールドには、以下の値を入れることができます。

**AP_NATIVE**
ネイティブ (APPN) 経路X定プロトコルのみをH用します。

**AP_NONNATIVE**
非ネイティブ (AnyNet) 経路X定プロトコルのみをH用します。

**AP_NATIVE_THEN_NONNATIVE**
ネイティブ (APPN) プロトコルをn行し、パートナー LU を見つけることができない場合は、非ネイティブ (AnyNet) プロトコルをH用してセッションh動化を再n行します。

**AP_NONNATIVE_THEN_NATIVE**
非ネイティブ (AnyNet) プロトコルをn行し、パートナー LU を見つけることができない場合は、ネイティブ (AnyNet) プロトコルをH用してセッションh動化を再n行します。

**注:** 後半の 3 つの値は、AnyNet DLC がノード・オペレーター機能でH用でき、かつ AnyNet リンク・ステーションが定義されている場合にのみ、意味をもちます。

**cp_create_parms.anynet_supported**
AnyNet DLC サポートのレベルをX定します。このフィールドは、以下のいずれかの値にすることができます。

**AP_NONE**
No ANYNET 機能がサポートされます。 **default_preference** フィールドには、AP_NATIVE 値をX定する , 要があります。

**AP_ACCESS_NODE**
非ネイティブ (AnyNet) 経路X定プロトコルのみをH用します。

**AP_NATIVE_THEN_NONNATIVE**
このノードは、ANYNET アクセス・ノード機能をサポートします。

**AP_GATEWAY**
このノードは、ANYNET ゲートウェイ機能を + Oします。この値は、**node_type** が AP_NETWORK_NODE に設定されている場合にのみ有効です。

**cp_create_parms.comp_in_series**
RLE 圧縮の前に LZ 圧縮がH用できるかどうかをX定します。

**AP_YES**

**AP_NO**

**cp_create_parms.max_ls_exception_events**
ノードによって記録される LS_EXCEPTION 項目の最大数をX定します。範囲は、0 ～ 200。

**cp_create_parms.max_compress_lvl**
ノードによってサポートされる最大圧縮レベル。

**AP_NONE**
ノードは圧縮をサポートしません。

**AP_RLE_COMPRESSION**
ノードは、LU 6.2 セッションでの RLE 圧縮と解凍、従来型 LU セッションでの RLE 圧縮と LZ9 解凍をサポートします。

**AP_LZ9_COMPRESSION**
ノードは、LZ9 および RLE 圧縮および解凍をサポートすることができます。

**AP_LZ10_COMPRESSION**
ノードは、LZ10、LZ9、および RLE 圧縮および解凍をサポートすることができます。

**AP_LZ12_COMPRESSION**
ノードは、LZ12、LZ10、LZ9、および RLE 圧縮および解凍をサポートすることができます。

**cp_create_parms.node_spec_data_len**
このフィールドは、常にゼロに設定しておかなければなりません。

**cp_create_parms.ptf**
将来のプログラム一~修正 (PTF) 操作を構成および制御するための配列。

**cp_create_parms.ptf[0]**
REQDISCONT サポート。パーソナル・コミュニケーションズまたは Communications Serverは、通常、REQDISCONT をH用して、セッション・トラフィックに , 要でなくなった限定 q 源ホスト・リンクを非 h 動化します。このバイトをH用すれば、パーソナル・コミュニケーションズまたは Communications Serverが REQDISCONT をH用するのを抑制したり、パーソナル・コミュニケーションズまたは Communications Serverによって送信される REQDISCONT 要求でH用された設定値を変更したりできます。

**AP_SUPPRESS_REQDISCONT**
このビットが設定されていれば、パーソナル・コミュニケーションズまたは Communications Serverは REQDISCOUNT をH用しません (このバイト内の他のすべてのビットは無 k されます)。

**AP_OVERRIDE_REQDISCONT**
このビットが設定されていれば、パーソナル・コミュニケーションズまたは Communications Serverは、以下の 2 つのビットに基づいて REQDISCOUNT の 8 準設定を上書きします。

**AP_REQDISCONT_TYPE**
このビットが設定されていれば、パーソナル・コミュニケーションズまたは Communications Serverは、REQDISCOUNT に “immediate” のタイプをX定します。このビットが設定されていなければ、パーソナル・コミュニケーションズまたは Communications Serverは、“normal” のタイプをX定します。 (AP_OVERRIDE_REQDISCONT が設定されなければ、このビットは無 k されます。)

**AP_REQDISCONT_RECONTACT**
このビットが設定されていれば、パーソナル・コミュニケーションズまたは Communications Serverは、 REQDISCOUNT に “immediate recontact” をX定します。このビットが設定されていなければ、パーソナル・コミュニケーションズまたは Communications Serverは、 “no immediate recontact” をX定します。 (AP_OVERRIDE_REQDISCONT が設定されなければ、このビットは無 k されます。)

**cp_create_parms.ptf[1]**
ERP サポート。

パーソナル・コミュニケーションズまたは Communications Serverは、通常、ACTPU(ERP) を ERP として処理します (ACTPU(ERP) は、PU_SSCP セッションのリセットを要求しますが、 ACTPU(cold) とは異なり、補助的な LU_SSCP および PLU_SLU セッションの暗黙的な非 h 動化を要求しません)。SNA のインプリメンテーションは、ACTPU(cold) と同様に、ACTPU(ERP) を正常処理することができます。

**AP_OVERRIDE_ERP**
このビットが設定されていれば、パーソナル・コミュニケーションズまたは Communications Server は、すべての ACTPU 要求を ACTPU(cold) として処理します。

**cp_create_parms.ptf[2]**
BIS サポート。

パーソナル・コミュニケーションズまたは Communications Serverは、通常、限定 q 源 LU 6.2 セッションを非 h 動化する前に BIS プロトコルをH用します。このバイトにより、BIS のH用の上書きが可能になります。

**AP_SUPPRESS_BIS**
このビットが設定されていれば、パーソナル・コミュニケーションズまたは Communications Server は BIS プロトコルをH用しません。限定 q 源 LU 6.2 セッションは、UNBIND (終結処理) をH用して即 ~ に非 h 動化されます。

**up_time**
ノードが + O（または再O動）されてから経過した ~ 間 (1/100 C単位)。

**mem_size**
基本オペレーティング・システムから記憶管理によって入手されたH用可能記憶域のサイズ。

**mem_used**
現在プロセスに d り振られている記憶域バイトの数。

**mem_warning_threshold**
d り振りしきい値。この値を超えると、記憶域管理は、記憶域 q 源が圧迫されていると見なします。

**mem_critical_threshold**
d り振りしきい値。この値を超えると、記憶域管理は、記憶域 q 源が極度に圧迫されていると見なします。

**nn_functions_supported**
予約済み。

**functions_supported**
どの機能がサポートされているかを示します。これは、以下の 1 つまたは複数の値になります。

AP_NEGOTIABLE_LS
AP_SEGMENT_REASSEMBLY
AP_BIND_REASSEMBLY

AP_PARALLEL_TGS
AP_CALL_IN
AP_ADAPTIVE_PACING
AP_TOPOLOGY_AWARENESS

**en_functions_supported**
サポートされているエンド・ノード機能をX定します。

**AP_SEGMENT_GENERATION**
ノードは、セグメント生成をサポートします。

**AP_MODE_TO_COS_MAP**
ノードは、モード名から COS 名へのマッピングをサポートします。

**AP_LOCATE_CDINIT**
ノードは、リモート LU を突き _ めるためのロケートの生成および定義域間 + O GDS 変数をサポートします。

**AP_REG_WITH_NN**
ノードは、自分の LU を隣接サービス・ネットワーク・ノードに登録します。

**AP_REG_CHARS_WITH_NN**
ノードは、登録特性送信をサポートします (登録名送信もサポートされている場合にのみサポートされます)。

**nn_status**
予約済み。

**nn_frsn**
予約済み。

**nn_rsn**
予約済み。

**def_ls_good_xids**
ノードが最後に + O されてから、すべての定義済みリンク・ステーションで行われた成功 XID 交換の合計数。

**def_ls_bad_xids**
ノードが最後に + O されてから、すべての定義済みリンク・ステーションで行われた失敗 XID 交換の合計数。

**dyn_ls_good_xids**
ノードが最後に + O されてから、すべての動的リンク・ステーションで行われた成功 XID 交換の合計数。

**dyn_ls_bad_xids**
ノードが最後に + O されてから、すべての動的リンク・ステーションで行われた失敗 XID 交換の合計数。

**dlur_release_level**
現行の DLUR リリース・レベルをX定します。

**nns_dlus_served_lu_reg_supp**
エンド・ノードのみ。エンド・ノードのネットワーク・ノード・サーバーが、 DLUS 提供の LU 登録をサポートするかどうかをX定します。

**AP_NO**
ネットワーク・ノード・サーバーは DLUS 提供の LU 登録をサポートしません。

**AP_YES**
ネットワーク・ノード・サーバーは DLUS 提供の LU 登録をサポートします。

**AP_UNKNOWN**
エンド・ノードはネットワーク・ノード・サーバーをもっていません。

NN のみ: このフィールドは AP_NO に設定されます。

**fq_nn_server_name**
17 バイトの長さをもつ、現行ネットワーク・ノード・サーバーの完全修飾名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

このノードがエンド・ノードでないか、アクティブ・ネットワーク・ノード・サーバーをもっていない場合は、このフィールドはヌルに設定されます。

**current_isr_sessions**
現在このノードを介して経路X定されているアクティブ ISR セッションの数。このノードがネットワーク・ノードでなければ、このフィールドはゼロに設定されます。

**nn_functions2**
サポートされているネットワーク・ノード機能をX定します。

**AP_BRANCH_AWARENESS**
ノードは ″ブランチ・アウェア″ です。

**branch_ntwk_arch_version**
サポートされているブランチ・ネットワーク・アーキテクチャーのバージョンをX定するか、またはノードがブランチ・ネットワーク・アーキテクチャーをサポートしていない場合は、ゼロをX定します。

**AP_BRANCH_AWARENESS**
ノードは ″ブランチ・アウェア″ です。

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_PARTNER_LU

QUERY_PARTNER_LU は、ローカル LU によってH用されているパートナー LU に関する情報を戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のパートナー LU に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **plu_alias** フィールドを設定する , 要があります (あるいは、**plu_alias** がすべてゼロに設定されている場合は、 **fqplu_name** を設定する , 要があります)。 **list_options** フィールドを AP_FIRST_IN_LIST に設定すれば、これらのフィールドの両方が無 k されます。 **lu_name** または **lu_alias** フィールドは、常に設定しておかなければなりません。**lu_name** が非ゼロであれば、 **lu_alias** よりも優先してH用されます。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**fqplu_name** 順に配列されます。配列はまず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、 ASCII の辞書配列の順序で行われます ( 8 準の MIB 配列に準拠)。 AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます (X定された項目が存在するしないに関係なく)。

**plu_alias** をすべてゼロに設定すると、 **fqplu_name** 値がH用されます。それ以 0 の場合は、**plu_alias** が常にH用され、 **fqplu_name** は無 k されます。

戻されたパートナー LU のリストは、現在これらの LU がアクティブ・セッションをもっているかどうかによってフィルター処理を行うことができます。フィルター処理を行いたい場合は、 **active_sessions** フィールドを AP_YES に設定する , 要があります (フィルター処理を行いたくない場合は、このフィールドを AP_NO に設定する , 要があります)。

この verb は、パートナー LU との間で少なくとも 1 つのセッションがN立されたときに決定された情報を戻します。

QUERY_PARTNER_LU_DEFINITION verb は、定義情報のみを戻します。

## VCB 構造体

```
typedef struct query_partner_lu
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   *buf_ptr;           /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this 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[RD_LEN]; /* resource description          */
        unsigned short  act_sess_count;      /* curr 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
{
        unsigned short  overlay_size;        /* size of this 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[RD_LEN]; /* resource description          */
        unsigned short  act_sess_count;      /* curr 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         */
        unsigned short  max_mc_ll_send_size; /* max send LL size for mapped   */
                                             /* conversations                 */
        unsigned char   implicit;            /* implicit or explicit entry    */
        unsigned char   security_details;    /* conversation security detail  */
        unsigned char   duplex_support;      /* full-duplex support           */
        unsigned char   preference;          /* routing preference            */
        unsigned char   reserva[16];         /* reserved                      */
} PLU_DETAIL;
```

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_PARTNER_LU

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンを X 定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーを X 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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

**lu_name** (または、 **lu_name** がすべてゼロに設定されている場合は、**lu_alias**)、および **plu_alias** (または、 **plu-alias** がすべてゼロに設定されている場合は、 **fqplu_name**) を組み合わせたX定 (以下のパラメーターを2照) は、戻された実際の情報の+O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
**plu_alias** および **fqplu_name** フィールドは無k され、戻りリストはリスト内の最初の項目から+Oされます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から+Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から+Oされます。

**lu_name**
LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文z列です。このフィールドをすべてゼロに設定すると、 **lu_alias** フィールドが索引の判別にH用されます。

**lu_alias**
ローカル定義の LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。このフィールドは、**lu_name** フィールドをすべてゼロに設定した場合にのみ有効です。この場合、8 バイトすべてが有効であるため、8 バイトすべてを設定する,要があります。**lu_name** と **lu_alias** を両方ともすべてゼロに設定すると、制御点と関連する LU (デフォルトの LU) がH用されます。

**plu_alias**
パートナー LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。8 バイトすべてが有効であり、すべて設定する,要があります。このフィールドをすべてゼロに設定すると、 **fqplu_name** フィールドが索引値としてH用されます。

**fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文z列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**active_sessions**
アクティブ・セッション・フィルター。戻されたパートナー LU を、現在これらの LU がアクティブ・セッションをもっているかどうかによってフィルター処理を行うかどうかをX定します (AP_YES または AP_NO)。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**plu_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**plu_summary.plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**plu_summary.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**plu_summary.description**
q 源の説明 (DEFINE_PARTNER_LU でＸ定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**plu_summary.act_sess_count**
ローカル LU とパートナー LU との間のアクティブ・セッションの合計数。**active_sessions** フィルターが AP_YES に設定されていれば、このフィルターは常にゼロよりも大きくなります。

**plu_summary.partner_cp_name**
パートナー LU の制御点の 17 バイト完全修飾ネットワーク名。この名前は、1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**plu_summary.partner_lu_located**
パートナー LU の制御点名が解決されたかどうかをＸ定します (AP_YES または AP_NO)。

**plu_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**plu_detail.plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**plu_detail.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**plu_detail.description**
q 源の説明 (DEFINE_PARTNER_LU で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**plu_detail.act_sess_count**
ローカル LU とパートナー LU との間のアクティブ・セッションの合計数。 **active_sessions** フィルターが AP_YES に設定されていれば、このフィルターは常にゼロよりも大きくなります。

**plu_detail.partner_cp_name**
パートナー LU の制御点の 17 バイト完全修飾ネットワーク名。この名前は、1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**plu_detail.partner_lu_located**
パートナー LU の制御点名が解決されたかどうかを X 定します (AP_YES または AP_NO)。

**plu_detail.plu_un_name**
パートナー LU の非解釈名。この名前は、8 バイト、タイプ A の EBCDIC 文 z 列です。

**plu_detail.parallel_sess_supp**
並列セッションがサポートされているかどうかを X 定します (AP_YES または AP_NO)。

**plu_detail.conv_security**
会話機密保護情報をこのパートナー LU に送信できるかどうかを X 定します (AP_YES または AP_NO)。 AP_NO に設定すると、トランザクション・プログラムによって提供されたどの機密保護情報もパートナー LU に送信されません。現在このパートナー LU とのアクティブ・セッションがない場合は、このフィールドは AP_UNKNOWN に設定されます。

**plu_detail.max_mc_ll_send_size**
パートナー LU に送信できる論理長 (LL) レコードの最大サイズ。これより大きいデータ・レコードは、いくつかの LL レコードに分けられてからパートナー LU に送信されます。 **max_mc_ll_send_size** が取り得る最大値は、32 767 です。

## QUERY_PARTNER_LU

**plu_detail.implict**

項目が、暗黙定義 (AP_YES) の結果であるか明示定義 (AP_NO) の結果であるかをX定します。

**plu_detail.security_details**

BIND で折衝された会話機密保護サポートを戻します。これは、以下の 1 つまたは複数の値にすることができます。

**AP_CONVERSATION_LEVEL_SECURITY**

会話機密保護情報は、会話をd り振るためのパートナー LU への要求またはパートナー LU からの要求に応じて、受け入られます。特定のタイプの会話機密保護サポートは、以下の値によって記述されます。

**AP_ALREADY_VERIFIED**

ローカル LU とパートナー LU の両方が、会話をd り振るための検査済みの要求を受け入れます。検査済みの要求では、ユーザー ID のみをX定し、パスワードはX定する，要はありません。

**AP_PERSISTENT_VERIFICATION**

}続検査は、ローカル LU とパートナー LU との間のセッションでサポートされます。つまり、会話に対する初期要求 (ユーザー ID をX定し、通常はパスワードもX定する) が検査されたら、会話に対する後続の要求では、ユーザー ID のみをX定する，要があります。

**AP_PASSWORD_SUBSTITUTION**

ローカル LU とパートナー LU は、パスワード置換会話機密保護をサポートします。会話をd り振るための要求が発行されるときは、その要求には暗号化形式のパスワードがX定されます。パスワード置換がサポートされない場合は、パスワードは、クリア・テキスト (暗号化されていない) 形式でX定されます。

**注:** セッションでパスワード置換がサポートされない場合は、 AP_PGM_STRONG の機密保護タイプをもつ ALLOCATE または SEND_CONVERSATION は失敗します。

**AP_UNKNOWN**

現在このパートナー LU とのアクティブ・セッションはありません。

**plu_detail.duplex_support**

BIND で折衝された会話二重サポートを戻します。以下のいずれかの値になります。

**AP_HALF_DUPLEX**

半二重会話のみがサポートされます。

**AP_FULL_DUPLEX**

全二重会話も半二重会話もサポートされます。

**AP_UNKNOWN**
パートナー LU とのアクティブ・セッションがないため、会話二重サポートが認識されません。

**plu_detail.preference**
DEFINE_PARTNER_LU verb にX定された経路X定プロトコル・プリファレンスを戻します。

**AP_NATIVE**
ネイティブ (APPN) 経路X定プロトコルのみをH用します。

**AP_NONNATIVE**
非ネイティブ (Anynet) プロトコルをH用し、パートナー LU を見つけることができない場合は、非ネイティブ (Anynet) プロトコルをH用してセッションの h 動化を再 n 行します。

**AP_NATIVE_THEN_NONNATIVE**
ネイティブ (APPN) プロトコルを n 行し、パートナー LU を見つけることができない場合は、ネイティブ (APPN) プロトコルをH用してセッションの h 動化を再 n 行します。

**AP_USE_DEFAULT_PREFERENCE**
ノードの + O ~ に定義されたデフォルトのプリファレンスをH用します。 (これは、START_NODE に設定され、QUERY_NODE で再呼び出しすることができます。)

非ネイティブ経路X定が意味をもつのは、 Anynet DLC が「プログラム」でH用でき、 Anynet リンク・ステーションが定義されている場合のみであることに注意してください。詳細については、 81ページの『DEFINE_LS』 を2照してください。

START_NODE で提供された **anynet_supported** フィールドを AP_NO に設定した場合は、このフィールドに値 AP_NATIVE または AP_USE_DEFAULT_PREFERENCE を入れなければなりません。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PLU_NAME

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_PARTNER_LU_DEFINITION

QUERY_PARTNER_LU_DEFINITION は、 DEFINE_PARTNER_LU verb ですでに渡されている情報を戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のパートナー LU に関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **plu_alias** フィールド (または、 **plu_alias**) がすべてゼロに設定されている場合は **fqplu_name** を設定する , 要があります。 **plu_alias** フィールドが非ゼロであれば、このフィールドは索引決定にH用され、**fqplu_name** は無 k されます。 **plu_alias** をすべてゼロに設定すると、 **fqplu_name** フィールドが索引決定にH用されます。 **list_options** フィールドを AP_FIRST_IN_LIST に設定すると、どちらのフィールドも無 k されます。(この場合、 AP_LIST_BY_ALIAS **list_options** が設定されていれば、戻りリストは **plu_alias** 順に配列されます。それ以 0 の場合は、**fqplu_name** 順に配列されます)。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、X定されたオプションに応じて、 **plu_alias** または **fqplu_name** のいずれかの順に配列されます。配列はまず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、 ASCII の辞書配列の順序で行われます ( 8 準の MIB 配列に準拠)。AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された順序に従って次の項目から + O されます (X定された項目が存在するかしないかに関係なく)。

この verb が定義情報のみを戻すことに注意してください。QUERY_PARTNER_LU verb は、パートナー LU との間で少なくとも 1 つのセッションがN立されるときに決定される情報を戻します。

## VCB 構造体

```
typedef struct query_partner_lu_definition
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   plu_alias[8];        /* partner LU alias              */
        unsigned char   fqplu_name[17];      /* fully qualified partner       */
                                             /* LU name                       */
        unsigned char   description[RD_LEN]; /* resource description          */
} PARTNER_LU_DEF_SUMMARY;
```

```
typedef struct partner_lu_def_detail
{
        unsigned short  overlay_size;        /* size of this 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[RD_LEN]; /* resource description          */
        unsigned char   plu_un_name[8];      /* partner LU uninterpreted name */
        unsigned char   preference;          /* routing preference            */
        unsigned short  max_mc_ll_send_size; /* max MC send LL size           */
        unsigned char   conv_security_ver;   /* already_verified accepted     */
        unsigned char   parallel_sess_supp;  /* parallel sessions supported?  */
        unsigned char   reserv2[8];          /* reserved                      */
} PLU_CHARS;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_PARTNER_LU_DEFINITION

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

X定された **plu_alias** (または、 **plu_alias** がすべてゼロに設定されている場合は **fqplu_name**) (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**AP_LIST_BY_ALIAS**
戻りリストは、**plu_alias** 順に配列されます。このオプションは、AP_FIRST_IN_LIST が X 定されている場合にのみ有効です。AP_LIST_FROM_NEXT または AP_LIST_INCLUSIVE を X 定すると、リストの配列は、**plu_alias** または **fqplu_name** のどちらが + O 点として提供されたかによって異なります。

**plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。このフィールドをすべてゼロに設定すると、 **fqplu_name** フィールドが , 須パートナー LU を X 定するために H 用されます。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。) このフィールドは、 **plu_alias** フィールドをすべてゼロに設定した場合にのみ有効です。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** よりも大きくすることができます。

**partner_lu_def_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**partner_lu_def_summary.plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**partner_lu_def_summary.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**partner_lu_def_summary.description**
q 源の説明 (DEFINE_PARTNER_LU でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**partner_lu_def_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**partner_lu_def_detail.plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効です。

**partner_lu_def_detail.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**partner_lu_def_detail.plu_chars.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**partner_lu_def_detail.plu_chars.plu_alias**
パートナー LU の別名。

**partner_lu_def_detail.plu_chars.description**
q 源の説明 (DEFINE_PARTNER_LU でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**partner_lu_def_detail.plu_chars.plu_un_name**
パートナー LU の非解釈名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。

**plu_chars.preference**
このパートナー LU とのセッションの h 動化のために優先H用される経路X定プロトコルの集合。このフィールドには、以下の値を入れることができます。

**AP_NATIVE**
ネイティブ (APPN) 経路X定プロトコルのみをH用します。

**AP_NONNATIVE**
非ネイティブ (AnyNet) 経路X定プロトコルのみをH用します。

**AP_NATIVE_THEN_NONNATIVE**
ネイティブ (APPN) プロトコルをn行し、パートナー LU を見つけることができない場合は、非ネイティブ (AnyNet) プロトコルをH用してセッションのh動化を再n行します。

**AP_NONNATIVE_THEN_NATIVE**
非ネイティブ (AnyNet) プロトコルをn行し、パートナー LU を見つけることができない場合は、ネイティブ (APPN) プロトコルをH用してセッションのh動化を再n行します。

**AP_USE_DEFAULT_PREFERENCE**
ノードの+O~に定義されたデフォルトのプリファレンスをH用します。

**注:** 非ネイティブ経路X定は、AnyNet DLC がノード・オペレーター機能でH用でき、かつ AnyNet リンク・ステーションが定義されている場合にのみ、意味をもちます。

**partner_lu_def_detail.plu_chars.max_mc_ll_send_size**
パートナー LU に送信できる論理長 (LL) レコードの最大サイズ。これより大きいデータ・レコードは、いくつかの LL レコードに分けられてからパートナー LU に送信されます。 **max_mc_ll_send_size** が取り得る最大値は、32 767 です。

**partner_lu_def_detail.plu_chars.conv_security_ver**
パートナー LU がローカル LU に代わって **user_ids** の妥当性を検査できるかどうか (つまり、パートナー LU が接続要求内に検査済み 8 識を設定できるかどうか) をX定します。

AP_YES
AP_NO

**partner_lu_def_detail.plu_chars.parallel_sess_supp**
並列セッションがサポートされているかどうかをX定します (AP_YES または AP_NO)。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PLU_NAME

AP_INVALID_LIST_OPTION

ノードが+Oされないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_PORT

QUERY_PORT は、ノードのポートに関する情報のリストを戻します。この情報は、『決定済みデータ』 (実行中に動的に収集されたデータ) および 『定義済みデータ』 (DEFINE_PORT のアプリケーションによって提供されたデータ) として構造化されます。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のポートに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **port_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、**port_name** 順に配列されます。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます (IBM の 6611 APPN MIB 配列に準拠)。AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された配列に従って、次の項目から + O されます ( X 定された項目が存在するしないに関係なく)。

戻されたポートのリストは、これらのポートが属している DLC の名前別にフィルター処理を行うことができます。この場合は、**dlc_name** フィールドを設定する , 要

```
        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                      */
        unsigned short  def_ls_good_xids;    /* number of successful XIDs     */
        unsigned short  def_ls_bad_xids;     /* number of unsuccessful XIDs   */
        unsigned short  dyn_ls_good_xids;    /* successful XIDs on dynamic    */
                                             /* LS count                      */
        unsigned short  dyn_ls_bad_xids;     /* failed XIDs on dynamic        */
        unsigned short  num_implicit_links;  /* number of implicit links      */
                                             /* active on this port           */
        unsigned char   neg_ls_supp;         /* are negotiable LSs supported? */
                                             /* LS count                      */
        unsigned char   abm_ls_supp;         /* are ABM LSs supported?        */
        unsigned long   start_time           /* start time                    */
        unsigned char   reserva[12];         /* reserved                      */
} PORT_DET_DATA;

typedef struct port_def_data
{
        unsigned char   description;         /* resource description          */
        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 links to EN are uplink*/
        unsigned char   reserv3[2];          /* NB_BYTE                       */
        unsigned long   port_number;         /* port number                   */
        unsigned short  max_rcv_btu_size;    /* max receive BTU size          */
        unsigned short  tot_link_act_lim;    /* total link activation limit   */
        unsigned short  inb_link_act_lim;    /* inbound link activation limit */
        unsigned short  out_link_act_lim;    /* outbound link activation limit */
        unsigned char   ls_role;             /* initial link station role     */
        unsigned char   retry_flags;         /* conditions for automatic retrys*/
                                             /* retries                       */
        unsigned short  max_activation_attempts;                             */
                                             /* how many automatic retries    */
        unsigned short  activation_delay_timer;
                                             /* delay between automatic retries*/
        unsigned char   reserv1[10];         /* reserved                      */
        unsigned char   implicit_dspu_template[8];
                                             /* implicit DSPU template        */
        unsigned short  implicit_ls_limit    /* max number of implicit links  */
        unsigned char   reserv2              /* reserved                      */
        unsigned char   implicit_dspu_services;
                                             /* implicit links support DSPUs  */
        unsigned short  implicit_deact_timer;
                                             /* Implicit link HPR link        */
                                             /* deactivation timer            */
        unsigned short  act_xid_exchange_limit;
                                             /* activation XID exchange limit */
        unsigned short  nonact_xid_exchange_limit;
                                             /* non-act. XID exchange limit   */
        unsigned char   ls_xmit_rcv_cap;     /* LS transmit-rcv capability    */
        unsigned char   max_ifrm_rcvd;       /* max number of I-frames that   */
                                             /* can be received               */
        unsigned short  target_pacing_count; /* target pacing count           */
        unsigned short  max_send_btu_size;   /* max send BTU size             */
        LINK_ADDRESS    dlc_data;            /* DLC data                      */
        LINK_ADDRESS    hpr_dlc_data;        /* HPR DLC data                  */
        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;
                                        /* Implicit links support         */
                                        /* HPR link-level error recovery  */
        unsigned char   retired1;       /* reserved                       */
        TG_DEFINED_CHARS default_tg_chars;  /* Default TG chars           */
        unsigned char   discovery_supported; /* Discovery function supported?  */
        unsigned short  port_spec_data_len;  /* length of port spec data       */
        unsigned short  link_spec_data_len;  /* length of link spec data       */
} PORT_DEF_DATA;

typedef struct link_address
{
        unsigned short  length;         /* length                         */
        unsigned short  reserve1;       /* reserved                       */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                        /* 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;    /* modem class                    */
        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 port_spec_data
{
            unsigned char  port_data[SIZEOF_PORT_SPEC_DATA];

}  PORT_SPEC_DATA;

typedef struct link_spec_data
{
            unsigned char  link_data[SIZEOF_LINK_SPEC_DATA];

}  LINK_SPEC_DATA;
```

## 指定Ｑｉａ－？－

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_PORT

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
これは、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

X定された **port_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O 点を X 定するために H 用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**port_name**
照会されるポートの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**dlc_name**
DLC 名フィルター。このフィールドは、すべてゼロに設定するか、またはローカル = 示可能文 z セットの 8 バイトの文 z 列に設定する , 要があります。このフィールドを設定すると、この DLC に属するポートのみが戻されます。すべてゼロに設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**port_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**port_summary.port_name**
このリンク・ステーションと関連するポートの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**port_summary.description**
q 源定義 (DEFINE_PORT で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**port_summary.port_state**
ポートの現行の状態を示します。

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

**port_summary.dlc_name**
DLC の名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**port_detail.overlay_size**
この項目内のバイト数 (**link_spec_data** を含む)。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**port_detail.port_name**
このリンク・ステーションと関連するポートの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**port_detail.det_data.port_state**
ポートの現行の状態を示します。

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

**port_detail.det_data.dlc_type**
DLC のタイプ。パーソナル・コミュニケーションズまたは Communications Serverは、以下のタイプをサポートします。

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

**port_detail.det_data.port_sim_rim**
初期設定モードの設定 (SIM) と初期設定モードの受信 (RIM) がサポートされているかどうかをX定します (AP_YES または AP_NO)。

**port_detail.det_data.def_ls_good_xids**
このポートが最後に + O されてから、このポートのすべての定義済みリンク・ステーションで行われた成功 XID 交換の合計数。

**port_detail.det_data.def_ls_bad_xids**
このポートが最後に + O されてから、このポートのすべての定義済みリンク・ステーションで行われた失敗 XID 交換の合計数。

**port_detail.det_data.dyn_ls_good_xids**
このポートが最後に + O されてから、このポートのすべての動的リンク・ステーションで行われた成功 XID 交換の合計数。

**port_detail.det_data.dyn_ls_bad_xids**
このポートが最後に + O されてから、このポートのすべての動的リンク・ステーションで行われた失敗 XID 交換の合計数。

**port_detail.det_data.num_implicit_links**
現在このポートでアクティブ状態になっている暗黙リンクの合計数。これには、動的リンク、およびディスカバリーの後に作成された暗黙リンクが含まれます。このポートでH用できるこのようなリンクの数は、PORT_DEF_DATA の **implicit_ls_limit** フィールドによって制限されます。

**port_detail.def_data.neg_ls_supp**
折衝可能リンク・ステーション AP_YES or AP_NO のためのサポート。

**port_detail.det_data.abm_ls_supp**
ABM リンク・ステーションのためのサポート。これは、DLC が + O されるまで認識されません。

AP_NO
AP_YES
AP_UNKNOWN

**port_detail.det_data.start_time**
ノードが + O されてから、このポートが最後に + O されるまでの経過 ~ 間 (1/100 C 単位で測定)。このポートが + O されると、ゼロがこのフィールドに戻されます。

**port_detail.def_data.description**
q 源定義 (DEFINE_PORT で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**port_detail.def_data.dlc_name**
関連する DLC の名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**port_detail.def_data.port_type**
ポートによって H 用された回線のタイプを X 定します。この値は、以下のいずれかの値と対応しています。

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

**port_detail.def_data.port_attributes[0]**
これはビット・フィールドです。このフィールドには、AP_NO 値または以下の値を入れることができます。

**AP_RESOLVE_BY_LINK_ADDRESS**
CONNECT_IN のリンク・アドレスを H 用し、次に、着呼を解決するための受信済み XID3 に X 定された CP 名 (またはノード ID) を H 用して、着呼を解決しようと n みることを X 定します。**port_type** フィールドが AP_PORT_SWITCHED に設定されていない限り、このビットは無 k されます。

**port_detail.def_data.implicit_uplink_to_en**
BrNN のみ: 隣接ノードがエンド・ノードである場合に、このポートから離れた暗黙リンク・ステーションがアップリンクであるかダウンリンクであるかを X 定します。このフィールドの値は、同一のパートナーとの既存のリンクがない場合にのみ考慮されます。それは、このようなリンクはまず、リンク・タイプの決定に H 用されるからです。

**AP_NO**
暗黙リンクはダウンリンクです。

**AP_YES**
暗黙リンクはアップリンクです。

その他のノード・タイプ: このフィールドは AP_NO に設定されます。

**port_detail.def_data.port_number**
ポート番号。

**port_detail.def_data.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**port_detail.def_data.tot_link_act_lim**
合計リンク h 動化限度。

**port_detail.def_data.inb_link_act_lim**
インバウンド・リンク h 動化限度。

**port_detail.def_data.out_link_act_lim**
アウトバウンド・リンク h 動化限度。

**port_detail.def_data.ls_role**
リンク・ステーション・ロール。このフィールドは、折衝可能 (AP_LS_NEG)、 1 次 (AP_LS_PRI)、または 2 次 (AP_LS_SEC) にすることができます。 **implicit_hpr_support** を AP_NO に設定すると、予約済みになります。

**port_detail.def_data.implicit_dspu_template**
DEFINE_DSPU_TEMPLATE verb で定義された DSPU テンプレートをX定します。このテンプレートは、ローカル・ノードが、このポートで h 動化された暗黙リンクに PU 集信を提供する場合の定義にH用されるものです。X定されたテンプレートがリンクの h 動化 ~ に存在しない (またはすでにそのインスタンス限度に達している) 場合は、 h 動化は失敗します。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する , 要があります。

**def_data.implicit_dspu_services** フィールドが AP_PU_CONCENTRATION に設定されていないと、このフィールドは予約済みになります。

**port_detail.def_data.implicit_ls_limit**
動的リンクおよびディスカバリーのために h 動化されたリンクも含め、このポートで同 ~ にアクティブ状態になりことができる暗黙リンク・ステーションの最大数をX定します。 0 の値は限度がないことを意味し、AP_NO_IMPLICIT_LINKS の値は暗黙リンクが許可されていることを意味します。

**def_data.implicit.dspu_services**
ローカル・ノードが、このポートで h 動化された暗黙リンクを介してダウンストリーム PU に提供するサービスをX定します。以下のいずれかの値に設定されます。

**AP_DLUR**
ローカル・ノードは、ダウンストリーム PU に DLUR サービスを提供します (DEFINE_DLUR_DEFAULTS verb によって構成された省略 ~ DLUS をH用)。この設定は、ローカル・ノードがネットワーク・ノードである場合にのみ有効です。

**AP_PU_CONCENTRATION**
ローカル・ノードは、ダウンストリーム PU に PU 集信を提供します (さらに、**def_data.implicit_dspu_template** フィールドにX定された DSPU テンプレートによってX定された定義をH用します)。

**AP_NONE**
ローカル・ノードは、このダウンストリーム PU にサービスを提供しません。

**port_detail.def_data.retry_flags**

AP_INHERIT_RETRY フラグが **def_data.retry_flags** の DEFINE_LS に設定されている場合に、このポートの h 動化が自動的に再 n 行される条件を X 定します。これはビット・フィールドであり、以下の値をビット単位で OR で結合した任意の値を取ることができます。

**AP_RETRY_ON_START**

リンクの h 動化を n 行しているときにリモート・ノードから応答がないと、 h 動化が再 n 行されます。 h 動化を n 行しているときに基本ポートが非アクティブ状態であると、「プログラム」はそれを h 動化しようとします。

**AP_RETRY_ON_FAILURE**

リンクがアクティブまたは保留アクティブ状態のときに失敗すると、リンクの h 動化が再 n 行されます。 h 動化を n 行しているときに基本ポートが失敗すると、「プログラム」はそれを h 動化しようとします。

**AP_RETRY_ON_DISCONNECT**

リンクがリモート・ノードによって正常停 _ されると、リンクの h 動化が再 n 行されます。

**AP_DELAY_APPLICATION_RETRIES**

アプリケーションによって + O された (START_LS またはオンデマンド・リンク h 動化を H 用) リンク h 動化再 n 行は、**activation_delay_timer** を H 用して歩調合わせされます。

**AP_DELAY_INHERIT_RETRY**

このフィールドのフラグで X 定された再 n 行条件のほかに、基本ポート定義の **retry_flags** フィールドに X 定された再 n 行条件も H 用されます。

**port_detail.def_data.max_activation_attempts**

少なくとも 1 つのフラグが **def_data.retry_flags** の DEFINE_LS に設定され、 DEFINE_LS の **def_data.max_activation_attempts** が AP_USE_DEFAULTS に設定されている場合を除き、このフィールドは効果を生じません。

このフィールドは、リモート・ノードが無応答の場合、または基本ポートが非アクティブ状態の場合に「プログラム」によって許容される再 n 行の回数を X 定します。この回数には、自動再 n 行とアプリケーション主導型の h 動化 n 行の両方の回数も含まれます。

この限度に達すると、自動再 n 行はこれ以上行われません。この条件は、STOP_LS、STOP_PORT、STOP_DLC、または成功した h 動化によってリセットされます。 START_LS または OPEN_LU_SSCP_SEC_RQ によって 1 回の h 動化 n 行が行われますが、 h 動化に失敗すると、再 n 行は行われません。

ゼロは ’限度がない’ ことを意味します。 AP_USE_DEFAULTS の値を X 定すると、 DEFINE_PORT で提供された **max_activiation_attempts** が H 用されます。

**ls_detail.def_data.activation_delay_timer**

少なくとも 1 つのフラグが **def_data.retry_flags** の DEFINE_LS に設定され、 DEFINE_LS の **def_data.max_activation_attempts** が AP_USE_DEFAULTS に設定されている場合を除き、このフィールドは効果を生じません。

このフィールドは、 AP_DELAY_APPLICATION_RETRIES ビットが **def_data.retry_flags** に設定されている場合に、「プログラム」が自動再 n 行間、およびアプリケーション主導型 h 動化 n 行間に待機する C 数を X 定します。

AP_USE_DEFAULTS の値を X 定すると、 DEFINE_PORT で提供された **activiation_delay_timer** が H 用されます。

ゼロを X 定すると、「プログラム」は 30 C 間、デフォルトのタイマーを H 用します。

**def_data.implicit_dspu_template**

DEFINE_DSPU_TEMPLATE verb で定義された DSPU テンプレートを X 定します。このテンプレートは、ローカル・ノードが、このポートで h 動化された暗黙リンクに PU 集信を提供する場合の定義に H 用されるものです。 X 定されたテンプレートがリンクの h 動化 ~ に存在しない (またはすでにそのインスタンス限度に達している) 場合は、 h 動化は失敗します。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する , 要があります。

**def_data.implicit_dspu_services** フィールドが AP_PU_CONCENTRATION に設定されていないと、このフィールドは予約済みになります。

**def_data.implicit.dspu_services**

ローカル・ノードが、このポートで h 動化された暗黙リンクを介してダウンストリーム PU に提供するサービスを X 定します。以下のいずれかの値に設定されます。

**AP_DLUR**

ローカル・ノードは、ダウンストリーム PU に DLUR サービスを提供します (DEFINE_DLUR_DEFAULTS verb によって構成された省略 ~ DLUS を H 用)。

**AP_PU_CONCENTRATION**

ローカル・ノードは、ダウンストリーム PU に PU 集信を提供します (さらに、**def_data.implicit_dspu_template** フィールドに X 定された DSPU テンプレートによって X 定された定義を H 用します)。

**AP_NONE**

ローカル・ノードは、このダウンストリーム PU にサービスを提供しません。

**def_data.implicit_deact_timer**

限定 q 源リンク非 h 動化タイマー (C 単位)。 **implicit_limited_resource** を AP_YES または AP_NO_SESSIONS に設定すると、データがこのタイマー設定 ~ 間内にリンクを通過せず、セッションがリンクを H 用しない場合、 HPR 可能な暗黙リンクは自動的に非 h 動化されます。

**implicit_limited_resource** を AP_INACTIVITY に設定すると、データがこのタイマーの設定～間内にリンクを通過しない場合、暗黙リンクは自動的に非 h 動化されます。

ゼロをX定すると、30 というデフォルト値がH用されます。それ以 0 の場合は、最小値は 5 です。 (これより小さい値を設定しても、X定して値は無 k され、5 がH用されます。) **implicit_limited_resource** を AP_NO に設定しない限り、このパラメーターは予約済みになることに注意してください。

**port_detail.def_data.act_xid_exchange_limit**
h 動化 XID 交換限度。

**port_detail.def_data.nonact_xid_exchange_limit**
非 h 動化 XID 交換限度。

**port_detail.def_data.ls_xmit_rcv_cap**
リンク・ステーションの伝送／受信機能をX定します。これは、双方向同～ (AP_LS_TWS) または双方向代替 (AP_LS_TWA) のどちらかです。

**port_detail.def_data.max_ifrm_rcvd**
肯定応答を送信する前に、ローカル・リンク・ステーションが受信できる I フレームの最大数。範囲: 1 ～ 127

**port_detail.def_data.target_pacing_count**
この TG の BIND 用として望ましいペーシング・ウィンドウを示す、 1 から 32 767 までの数値。この数値は、固定バインド・ペーシングが実行される場合にのみ有効です。パーソナル・コミュニケーションズまたは Communications Serverは現在この値をH用していません。

**port_detail.def_data.max_send_btu_size**
送信可能な BTU の最大サイズ。

**port_detail.def_data.dlc_data.length**
ポート・アドレスの長さ。

**port_detail.def_data.dlc_data.address**
ポート・アドレス。

**port_detail.def_data.hpr_dlc_data.length**
HPR ポート・アドレスの長さ。

**port_detail.def_data.hpr_dlc_data.address**
HPR ポート・アドレス。これは、現在、HPR リンクがサポートされている場合にH用されます。このフィールドは、パーソナル・コミュニケーションズまたは Communications Serverから、このポートをH用しているリンク・ステーションで交換された XID3 の X'61' 制御ベクトルの X'80' サブフィールドに送信された情報をX定します。

**port_detail.def_data.implicit_cp_cp_sess_support**
このポートから離れた暗黙リンク・ステーションで CP-CP セッションを実行できるかどうかをX定します (AP_YES または AP_NO)。

**port_detail.def_data.implicit_limited_resource**
リンクをH用しているセッションがないときに、このポートから離れた暗黙リンク・ステーションを非 h 動化する，要があるかどうかをX定します。以下のいずれかの値に設定されます。

**AP_NO**
暗黙リンクは限定 q 源ではなく、自動的に非 h 動化されることはありません。

**AP_YES or AP_NO_SESSIONS**
暗黙リンクは限定 q 源であり、これらのリンクをH用しているアクティブ・セッションがないときに自動的に非 h 動化されます。

**AP_INACTIVITY**
暗黙リンクは限定 q 源であり、そのリンクをH用しているアクティブ・セッションがないとき、もしくは **implicit_deact_timer** フィールドでX定した~間内にリンクでの後続データがないときに、暗黙リンクは自動的に非 h 動化されます。

**port_detail.def_data.implicit_hpr_support**
HPR が暗黙リンクでサポートされるかどうかをX定します (AP_YES または AP_NO)。

**port_detail.def_data.implicit_link_lvl_error**
リンク・レベル・エラー回 | をH用して、HPR トラフィックが暗黙リンクで送信されるかどうかをX定します (AP_YES または AP_NO)。

**port_detail.def_data.default_tg_chars**
TG 特性 (37ページの『DEFINE_COS』 を 2 照)。この特性は、このポートから離れた暗黙リンク・ステーションにH用されるほか、**use_default_tg_chars** をX定した定義済みリンク・ステーションにもH用されます。

**port_detail.def_data.discovery_supported**
ディスカバリー検索機能がこのポート上で実行されるかどうかをX定します (AP_YES または AP_NO)。

**port_detail.def_data.port_spec_data_len**
ACTIVATE_PORT シグナル上のポートに未変更のまま渡されたデータの、埋め込みスペースを含まない長さ (バイト数)。このデータは、PORT_DETAIL 構造体に連結されます。

**port_detail.def_data.link_spec_data_len**
初期設定~に、リンク・ステーション構成要素に未変更のまま渡されたデータ。このデータは、ポート固有のデータのすぐ後の PORT_DETAIL 構造体に連結されます。ポート固有のデータとリンク固有のデータを組み合わせて、 4 バイト境&の終わりまで埋め込みます。ポート固有のデータとリンク固有のデータとの間には、明示的な埋め込みはありません。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PORT_NAME

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_PU

QUERY_PU は、ローカル PU およびそれと関連するリンクのリストを戻します。

この情報はリストとして戻されます。特定の PU に関する情報または いくつかの『固まり』 に分けられたリスト情報を入手するには、 **pu_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

この verb は、ローカル PU がホスト・システムに直接接続されるか、または DLUR を介して接続されるかを X 定します。 **host_attachment** フィールドをフィルターとして H 用して、X 定された接続タイプに関する情報だけを戻すことができます。

## VCB 構造体

```
typedef struct query_pu
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* Verb attributes               */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  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               */
} QUERY_PU;

typedef struct pu_data
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   pu_name[8];          /* PU name                       */
        unsigned char   description[RD_LEN]; /* resource description          */
        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;
                                             /* Data compression requested    */
                                             /* for conventional LU sessions  */
        unsigned char   conventional_lu_cryptography;
                                             /* Cryptography required for     */
                                             /* conventional LU sessions      */
        unsigned char   reserva[12];         /* reserved                      */
} PU_DATA;

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size       */
        unsigned short  send_ru_size;        /* session send RU size          */
        unsigned short  max_send_btu_size;   /* max send BTU size             */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size              */
        unsigned short  max_send_pac_win;    /* max send pacing window size   */
        unsigned short  cur_send_pac_win;    /* curr send pacing window size  */
        unsigned short  max_rcv_pac_win;     /* max recv pacing window size   */
        unsigned short  cur_rcv_pac_win;     /* current receive pacing        */
```

```
                                        /* window size                    */
        unsigned long   send_data_frames;    /* number of data frames sent     */
        unsigned long   send_fmd_data_frames;
                                        /* num of FMD data frames sent    */
        unsigned long   send_data_bytes;     /* number of data bytes sent      */
        unsigned long   rcv_data_frames;     /* num data frames received       */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames rcvd    */
        unsigned long   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;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_PU

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをXすポインター。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

これは、リスト情報に何を戻すかを示します。

Xされた **pu_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**pu_name**
リストされる最初の PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。 **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**host_attachment**
ホスト処理装置接続機構のためのフィルター。

**AP_NONE**
すべてのローカル PU に関する情報を戻します。

**AP_DLUR_ATTACHED**
DLUR によってサポートされるすべてのローカル PU に関する情報を戻します。

**AP_DIRECT_ATTACHED**
ホスト・システムに直接接続される PU に関する情報のみを戻します。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**pu_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**pu_data.pu_name**
PU 名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**pu_data.description**
q 源の説明 (DEFINE_LS または DEFINE_INTERNAL_PU で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**pu_data.ls_name**
この PU と関連するリンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**pu_data.pu_sscp_sess_active**
PU-SSCP セッションがアクティブであるかどうかを X 定します (AP_YES または AP_NO)。

**pu_data.host_attachment**
ローカル PU ホスト処理装置接続機構のタイプ。

**AP_DLUR_ATTACHED**
PU は、DLUR によってホスト・システムと接続されています。

**AP_DIRECT_ATTACHED**
PU は直接ホスト・システムと接続されます。

**pu_data.pu_sscp_stats.rcv_ru_size**
このフィールドは常に予約済みです。

**pu_data.pu_sscp_stats.send_ru_size**
このフィールドは常に予約済みです。

**pu_data.pu_sscp_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**pu_data.pu_sscp_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**pu_data.pu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_data.pu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_data.pu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_data.pu_sscp_stats.cur_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_data.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_data.pu_sscp_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**pu_data.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_data.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_data.pu_sscp_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**pu_data.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_data.pu_sscp_stats.sidh**
セッション ID 上位バイト。

**pu_data.pu_sscp_stats.sidl**
セッション ID 下位バイト。

**pu_data.pu_sscp_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、ACTPU の送信側はこのフィールドをゼロに設定し、 ACTPU の送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドを 1 に設定します。

**pu_data.pu_sscp_stats.ls_name**
統計と関連するリンク・ステーション名。 これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**pu_data.pu_sscp_stats.pacing_type**
PU-SSCP セッションでH用される受信ペーシング・タイプ。このフィールドには、AP_NONE 値が入ります。

**sscp_id**
これは 6 バイトのフィールドで、このフィールドには、この LU でH用された PU の ACTPU で受信された SSCP ID が含まれています。

**lu_sscp_sess_active** が AP_YES でなければ、このフィールドはゼロになります。

**pu_data.conventional_lu_compression**
この PU をH用するセッションにデータ圧縮を要求するかどうかをX定します。

**AP_NO**
ローカル・ノードは、この PU をH用するセッションを流れるデータを圧縮または圧縮解除してはなりません。

**AP_YES**
ホストがデータ圧縮を要求した場合は、この PU に従属するセッションに対して、データ圧縮をH用可能にする , 要があります。

**pu_data.conventional_lu_cryptography**
この PU に従属する従来型の LU セッションにセッション・レベル暗号化が , 要であるかどうかをX定します。

**AP_NONE**
「プログラム」は、セッション・レベル暗号化を行いません。

**AP_MANDATORY**
LU がインポート・キーをH用できる場合、「プログラム」は、 , 須のセッション・レベル暗号化を実行します。そうでない場合は、LUをH用するアプリケーションによってそれを実行しなければなりません (これが PU 集信であれば、それはダウンストリーム LU によって実行されます)。

**AP_OPTIONAL**
この値は、H 用する暗号化が、セッションごとにホスト・アプリケーションによって起動されるようにします。ホストが、この PU に従属するセッションの暗号化を要求した場合は、「プログラム」は AP_MANDATORY の場合のような行動を取ります。ホストが暗号化を要求しなかった場合は、その行動は AP_NONE と同じになります。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_RTP_CONNECTION

QUERY_RTP_CONNECTION は、ネットワーク・ノードまたはエンド・ノードでH用され、ノードがエンドポイントとなっている高速トランスポート・プロトコル (RTP) 接続に関するリスト情報を戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定の RTP 接続に関する情報またはいくつかの 『固まり』 にわけられたリスト情報を入手するには、 **rtp_name** フィールドを設定する , 要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式のH用方法に関する 2 考情報については、 10 ページの『ノードの照会』 を 2 照してください。

このリストは、 **rtp_name** 順に配列されます。配列は、まず、名前の長さ順に行われ、次に、名前の長さが同じ場合は、ASCII 辞書配列の順序で行われます ( 8 準の MIB 配列に準拠)。AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された順序に従って次の項目から + O されます (X定された項目が存在するかしないかに関係なく)。

## VCB 構造体

```
typedef struct query_rtp_connection
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* Primary return code            */
        unsigned long   secondary_rc;        /* Secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this 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   reserv1;             /* reserved                       */
        unsigned char   cos_name[8];         /* class-of-service name          */
        unsigned short  num_sess_active;     /* number of active sessions      */
} RTP_CONNECTION_SUMMARY;

typedef struct rtp_connection_detail
{
        unsigned short  overlay_size;        /* size of this 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;     /* connection provides ISR BF     */
        unsigned char   reserv1[3];          /* reserved                       */
```

```
        unsigned char   cos_name[8];         /* class-of-service name          */
        unsigned short  max_btu_size;        /* max BTU size                   */
        unsigned long   liveness_timer;      /* liveness timer                 */
        unsigned char   local_tcid[8];       /* local TCID                     */
        unsigned char   remote_tcid[8];      /* remote TCID                    */
        RTP_STATISTICS  rtp_stats;           /* RTP statistics                 */
        unsigned short  num_sess_active;     /* number of active sessions      */
        unsigned char   reserv2[16];         /* reserved                       */
        unsigned short  rscv_len;            /* length of appended RSCV        */
} RTP_CONNECTION_DETAIL;

typedef struct rtp_statistics
{
        unsigned long   bytes_sent;          /* total number of bytes sent     */
        unsigned long   bytes_received;      /* total number of bytes received */
        unsigned long   bytes_resent;        /* total number of bytes resent   */
        unsigned long   bytes_discarded;     /* total number bytes discarded   */
        unsigned long   packets_sent;        /* total number of packets sent   */
        unsigned long   packets_received;    /* total number packets received  */
        unsigned long   packets_resent;      /* total number of packets resent */
        unsigned long   packets_discarded;   /* total number packets discarded */
        unsigned long   gaps_detected;       /* gaps detected                  */
        unsigned long   send_rate;           /* current send rate              */
        unsigned long   max_send_rate;       /* maximum send rate              */
        unsigned long   min_send_rate;       /* minimum send rate              */
        unsigned long   receive_rate;        /* current receive rate           */
        unsigned long   max_receive_rate;    /* maximum receive rate           */
        unsigned long   min_receive_rate;    /* minimum receive rate           */
        unsigned long   burst_size;          /* current burst size             */
        unsigned long   up_time;             /* total uptime of connection     */
        unsigned long   smooth_rtt;          /* smoothed round-trip time       */
        unsigned long   last_rtt;            /* last round-trip time           */
        unsigned long   short_req_timer;     /* SHORT_REQ timer duration       */
        unsigned long   short_req_timeouts;  /* number of SHORT_REQ timeouts   */
        unsigned long   liveness_timeouts;   /* number of liveness timeouts    */
        unsigned long   in_invalid_sna_frames;
                                             /* number of invalid SNA frames   */
                                             /* received                       */
        unsigned long   in_sc_frames;        /* number of SC frames received   */
        unsigned long   out_sc_frames;       /* number of SC frames sent       */
        unsigned char   reserve[40];         /* reserved                       */
} RTP_STATISTICS;
```

**注: rtp_connection_detail** オーバーレーの後に、 SNA によって定義されている経路選択制御ベクトル (RSCV) が続きます。 RTP 接続の設定から任意のパス切り替えまでの間、以下のように RTP 接続の RSCV が F ノードで J 納され、 = 示されます。

- RSCV には、ローカル・ノードからパートナー RTP ノードへのすべてのホップが入れられます。
- パートナー RTP ノードが、RTP 接続を h 動化させるセッションのエンドポイントでなければ、 RSCV は、パートナー RTP ノードから切り離す 1 つの “境&機能ホップ” も J 納します。
- ローカル・ノードにセッション・エンドポイントが含まれていない場合でも、 RSCV には、ローカル・ノード内に導く境&機能ホップは入れられません。

パス切り替えが終わると、保管され = 示された RSCV には、ローカル・ノードからパートナー RTP ノードへのホップのみが入れられます。(境&機能ホップは入れられません)。

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**
AP_QUERY_RTP_CONNECTION

**format**
VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**
提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**
これは、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

X定された **rtp_name** は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
**rtp_name** は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**rtp_name**
RTP 接続名。この名前は、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**rtp_connection_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**rtp_connection_summary.rtp_name**
RTP 接続名。この名前は、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**rtp_connection_summary.first_hop_ls_name**
RTP 接続の最初のホップのリンク・ステーション名。この名前は、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**rtp_connection_summary.dest_node_name**
RTP 接続の 17 バイトの完全修飾名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**rtp_connection_summary.cos_name**
RTP 接続のサービス・クラス名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列で、右側に EBCDIC スペースが埋め込まれています。

**rtp_connection_summary.num_sess_active**
RTP 接続上で現在アクティブ状態になっているセッションの数。

**rtp_connection_detail.overlay_size**
この項目内のバイトの数 (U加 RSCV を含む)。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**rtp_connection_detail.rtp_name**
RTP 接続名。この名前は、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**rtp_connection_detail.first_hop_ls_name**
RTP 接続の最初のホップのリンク・ステーション名。この名前は、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**rtp_connection_detail.dest_node_name**
RTP 接続の宛先ノードの 17 バイトの完全修飾名。 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**rtp_connection_detail.isr_boundary_fn**
ローカル・ノードによって HPT-APPN 境&機能が提供されるすべての ISR セッションに RTP 接続を H 用する場合は AP_YES。それ以 0 の場合は、AP_NO。

**rtp_connection_detail.cos_name**
RTP 接続のサービス・クラス名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列で、右側に EBCDIC スペースが埋め込まれています。

**rtp_connection_detail.max_btu_size**
RTP 接続の最大 BTU サイズ (バイト単位)。

**rtp_connection_detail.liveness_timer**
RTP 接続の h 性タイマー (C 単位)。

**rtp_connection_detail.local_tcid**
RTP 接続のローカル TCID。

**rtp_connection_detail.remote_tcid**
RTP 接続のリモート TCID。

**rtp_connection_detail.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_connection_detail.rtp_stats.bytes_received**
ローカル・ノードがこの RTP 接続上で受信したバイトの合計数。

**rtp_connection_detail.rtp_stats.bytes_resent**
転送中のロスのためにローカル・ノードが再送したバイトの合計数。

**rtp_connection_detail.rtp_stats.bytes_discarded**
RTP 接続のもう一方の終端によって送信されたバイトのうち、すでに受信されたデータの重複として廃棄されたバイトの合計数。

**rtp_connection_detail.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。

**rtp_connection_detail.rtp_stats.packets_received**
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_connection_detail.rtp_stats.packets_resent**
転送中のロスのためにローカル・ノードが再送したパケットの合計数。

**rtp_connection_detail.rtp_stats.packets_discarded**
RTP 接続のもう一方の終端によって送信されたパケットのうち、すでに受信したデータの重複として破棄されたパケットの合計数。

**rtp_connection_detail.rtp_stats.gaps_detected**
ローカル・ノードによって検出されたギャップの合計数。それぞれのギャップは、1 つまたは複数の消失フレームに対応しています。

**rtp_connection_detail.rtp_stats.send_rate**
この RTP 接続での現行送信速度 (キロビット/C)。これは、ARB アルゴリズムによって ; 出される最大許容送信速度です。

**rtp_connection_detail.rtp_stats.max_send_rate**
この RTP 接続での最大送信速度 (キロビット/C)。

**rtp_connection_detail.rtp_stats.min_send_rate**
この RTP 接続での最小送信速度 (キロビット/C)。

**rtp_connection_detail.rtp_stats.receive_rate**
この RTP 接続での現行受信速度 (キロビット/C)。これは、最後の測定間V で計；された実際の受信速度です。

**rtp_connection_detail.rtp_stats.max_receive_rate**
この RTP 接続での最大受信速度 (キロビット/C)。

**rtp_connection_detail.rtp_stats.min_receive_rate**
この RTP 接続での最小受信速度 (キロビット/C)。

**rtp_connection_detail.rtp_stats.burst_size**
RTP 接続での現行バースト・サイズ (バイト単位)。

**rtp_connection_detail.rtp_stats.up_time**
RTP 接続がアクティブ状態になっている合計C数。

**rtp_connection_detail.rtp_stats.smooth_rtt**
ローカル・ノードとパートナー RTP ノードとの間の往 | 接続 ~ 間の平 j 化測定 (ミリC単位)。

**rtp_connection_detail.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、最後に測定された往 | ~ 間 (ミリC単位)。

**rtp_connection_detail.rtp_stats.short_req_timer**
SHORT_REQ タイマーにH用される現行所要 ~ 間 (ミリC単位)。

**rtp_connection_detail.rtp_stats.short_req_timeouts**
この RTP 接続で SHORT_REQ タイマーの有効期限が切れた回数の合計数。

**rtp_connection_detail.rtp_stats.liveness_timeouts**
この RTP 接続で h 性タイマーの有効期限が切れた回数の合計数。 h 性タイマーの有効期限が切れるのは、 **rtp_connection_detail.liveness_timer** でX定した期間、接続がアイドル状態になったときです。

**rtp_connection_detail.rtp_stats.in_invalid_sna_frames**
この RTP 接続上で受信された SNA フレームのうち、無効であるとして破棄されたものの合計数。

**rtp_connection_detail.rtp_stats.in_sc_frames**
この RTP 接続上で受信されたセッション制御フレームの合計数。

**rtp_connection_detail.rtp_stats.out_sc_frames**
この RTP 接続上で送信されたセッション制御フレームの合計数。

**rtp_connection_detail.num_sess_active**
この RTP 接続上で現在アクティブ状態になっているセッションの数。

**rtp_connection_detail.rscv_len**
RTP 接続のU加ルート選択制御ベクトルの長さ。 (何もU加されなかった場合、長さはゼロになります。) RSCV は、次の詳細項目を正しく位置合わせできるように 4 バイト境& に合わせて終わりまで埋め込みまれますが、**rscv_len** にはこの埋め込みは含まれません。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_RTP_CONNECTION

AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_SESSION

QUERY_SESSION は、ノードがエンドポイントになっているセッションに関するリスト情報を戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のセッションに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **session_id** フィールドを設定する，要があります。そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。 **lu_name** (または **lu_alias**) および **plu_alias** (または **fqplu_name**) フィールドを、常に設定しておく，要があることに注意してください。**lu_name** が非ゼロであれば、 **lu_alias** よりも優先して H 用されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

戻されたセッションのリストは、パートナー LU の名前別にフィルター処理を行うことができます。これを行うには、**fqplu_name** または **plu_alias** フィールドを設定する，要があります。 **plu_alias** をすべてゼロに設定すると、**fqplu_name** 値が H 用されます。そうでない場合は、**plu_alias** が常に H 用され、**fqplu_name** は無 k されます。

戻されたセッションのリストは、これらのセッションが関連しているモードの名前別にフィルター処理を行うことができます。この場合は、 **mode_name** フィールドを設定する，要があります (フィルター処理を行わない場合は、このフィールドをすべてゼロに設定する，要があります)。

START NODE パラメーターに経路選択制御ベクトル (RSCV) を X 定すると、F セッションに関する詳細情報のほかに、RSCV も戻されます (キー長形式で)。この RSCV (*Sna Formats* で X 定) は、セッションがホップ・バイ・ホップ形式で H 用するネットワーク内の経路を定義します。

## VCB 構造体

```
typedef struct query_session
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   *buf_ptr;           /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this 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
{
        unsigned short  overlay_size;        /* size of this 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   duplex_support;      /* full-duplex support           */
        unsigned char   sscp_id[6];          /* SSCP ID of host               */
        unsigned char   reserva[20];         /* reserved                      */
        unsigned long   session_start_time;  /* start time of the session     */
        unsigned short  session_timeout;     /* session timeout               */
        unsigned char   reservb[7];          /* 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;

typedef struct fqpcid
{
        unsigned char   pcid[8];             /* pro correlator identifier     */
        unsigned char   fqcp_name[17];       /* orig's network qualified      */
                                             /* CP name                       */
        unsigned char   reserve3[3];         /* reserved                      */
} FQPCID;

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size       */
        unsigned short  send_ru_size;        /* session send RU size          */
        unsigned short  max_send_btu_size;   /* Maximum send BTU size         */
        unsigned short  max_rcv_btu_size;    /* Maximum rcv BTU size          */
        unsigned short  max_send_pac_win;    /* Max send pacing window size   */
        unsigned short  cur_send_pac_win;    /* Curr send pacing window size  */
        unsigned short  max_rcv_pac_win;     /* Max receive pacing win size   */
        unsigned short  cur_rcv_pac_win;     /* Curr rec pacing window size   */
        unsigned long   send_data_frames;    /* Number of data frames sent    */
        unsigned long   send_fmd_data_frames;
                                             /* num of FMD data frames sent   */
        unsigned long   send_data_bytes;     /* Number of data bytes sent     */
        unsigned long   rcv_data_frames;     /* Num data frames received      */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames recvd  */
        unsigned long   rcv_data_bytes;      /* Num 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[8];         /* Link station name            */
        unsigned char   pacing_type;        /* type of pacing in use        */
} SESSION_STATS;
```

**注:** セッション詳細オーバーレーの後に、 *SNA 形式* によって定義された経路選択制御ベクトル (RSCV) が続くことがあります。この制御ベクトルは、ネットワーク内のセッション経路を定義し、BIND で送信されます。 START_NODE verb のフィールドを AP_YES に設定すると、RSCV が組み込まれます (キー長形式で)。START_NODE verb のフィールドを AP_NO に設定すると、 **rscv_len** はゼロに設定されます。

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_SESSION

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

これは、リスト情報に何を戻すかを示します。

**AP_SUMMARY**

要約情報のみを戻します。

**AP_DETAIL**

詳細情報を戻します。

**lu_name** (または、**lu_name** がすべてゼロに設定されている場合は **lu_alias**)、 **pu_alias** (または **plu_alias** がすべてゼロに設定されている場合は **fqplu_name**)、 **mode_name**、および **session_id** を組み合わせたX定 (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

**session_id** は無 k され、戻りリストは、リスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + Oされます。

**lu_name**
LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文z列です。このフィールドをすべてゼロに設定すると、 **lu_alias** フィールドが索引の判別にH用されます。

**lu_alias**
ローカル定義 LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。このフィールドは、**lu_name** フィールドをすべてゼロに設定した場合にのみ有効です。この場合、8 バイトすべてが有効であるため、8 バイトすべてを設定する,要があります。 **lu_name** および **lu_alias** の両フィールドをすべてゼロに設定すると、制御点 (デフォルト LU) と関連する LU がH用されます。

**plu_alias**
パートナー LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。8 バイトすべてが有効であり、すべて設定する,要があります。このフィールドをすべてゼロに設定すると、 **fqplu_name** フィールドが索引値の判別にH用されます。

**fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文z列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**mode_name**
モード名フィルター。このフィールドをすべてゼロに設定するか、 8 バイト英数zのタイプ A の EBCDIC 文z列に設定し (文zでOまる)、右側に EBCDIC スペースを埋め込む,要があります。このフィールドを設定すると、このモードと関連するセッションのみが戻されます。すべてゼロに設定すると、このフィールドは無kされます。

**session_id**
8 バイトのセッション ID。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無kされます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**session_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**session_summary.plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**session_summary.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。

**session_summary.mode_name**
モード名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で Oまる) で、右側に EBCDIC スペースが埋め込まれています。

**session_summary.session_id**
8 バイトのセッション ID。

**session_summary.fqpcid.pcid**
プロシージャー相関関係R ID。これは 8 バイトの 16 進文 z 列です。

**session_summary.fqpcid.fqcp_name**
完全修飾制御点名。この 17 バイトの名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**session_detail.overlay_size**
この項目内のバイトの数 (U加 RSCV を含む)。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**session_detail.plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**session_detail.fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。

**session_detail.mode_name**
モード名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で Oまる) で、右側に EBCDIC スペースが埋め込まれています。

**session_detail.session_id**
8 バイトのセッション ID。

**session_detail.fqpcid.pcid**
プロシージャー相関関係R ID。これは 8 バイトの 16 進文 z 列です。

**session_detail.fqpcid.fqcp_name**
完全修飾制御点名。この 17 バイトの名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。 (それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**session_detail.cos_name**
サービス・クラス名。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z でOまる) で、右側に EBCDIC スペースが埋め込まれています。

**session_detail.trans_pri**
伝送優先順位。以下のいずれかの値に設定されます。

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

**session_detail.ltd_res**
セッションが、限定 q 源リンクをH用するかどうかを示します (AP_YES または AP_NO)。

**session_detail.polarity**
セッションの極性をX定します (AP_PRIMARY または AP_SECONDARY)。

**session_detail.contention**
セッション競合極性をX定します。これは、ローカル LU がこのセッションのH用に対する ’第 1 拒否権’ をもっているか (AP_CONWINNER)、またはセッションをH用する前に送信権を要求しなければならないか (AP_CONLOSER) を示します。

**session_detail.sess_stats.rcv_ru_size**
受信 RU の最大サイズ。

**session_detail.sess_stats.send_ru_size**
最大送信 RU サイズ。

**session_detail.sess_stats.max_send_btu_size**
送信可能な BTU の最大サイズ。

**session_detail.sess_stats.max_rcv_btu_size**
受信可能な BTU の最大サイズ。

**session_detail.sess_stats.max_send_pac_win**
このセッションでの送信ペーシング・ウィンドウの最大サイズ。

**session_detail.sess_stats.cur_send_pac_win**
このセッションでの送信ペーシング・ウィンドウの現行サイズ。

**session_detail.sess_stats.max_rcv_pac_win**
このセッションでの受信ペーシング・ウィンドウの最大サイズ。

**session_detail.sess_stats.cur_rcv_pac_win**
このセッションでの受信ペーシング・ウィンドウの現行サイズ。

**session_detail.sess_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**session_detail.sess_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**session_detail.sess_stats.send_data_bytes**
送信された通常フロー・データ・バイトの数。

**session_detail.sess_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**session_detail.sess_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**session_detail.sess_stats.rcv_data_bytes**
受信された通常フロー・データ・バイトの数。

**session_detail.sess_stats.sidh**
セッション ID 上位バイト。

**session_detail.sess_stats.sidl**
セッション ID 下位バイト。

**session_detail.sess_stats.odai**
起点宛先アドレス 8 識。セッション + O ~ に、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れば、BIND の送信側はこのフィールドをゼロに設定します。 BIND 送信側が 2 次リンク・ステーションが含まれているノードであれば、このフィールドは 1 に設定されます。

**session_detail.sess_stats.ls_name**
統計と関連するリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。このフィールドを H 用して、セッション統計と、セッション・データが流れるリンクとを関連 U けることができます。

**session_detail.sess_stats.pacing_type**
このセッションで H 用される受信ペーシング。このフィールドには、AP_NONE、AP_PACING_FIXED、または AP_PACING_ADAPTIVE 値を入れることができます。

**session_detail.duplex_support**
BIND で折衝された会話二重サポートを戻します。以下のいずれかの値になります。

**AP_HALF_DUPLEX**
半二重会話だけがサポートされます。

**AP_FULL_DUPLEX**
全二重会話も半二重会話もサポートされます。優先データもサポートされます。

**session_detail.sscp_id**
従属 LU セッションの場合、このフィールドには、ローカル LU がマップさ

れた PU のホストから ACTPU に受信した SSCP ID が含まれています。独立 LU セッションの場合は、このフィールドはすべて 2 進ゼロに設定されます。

**session_detail.session_start_time**
CP が + O されてからこのセッションがアクティブになるまでの経過 ~ 間 (1/100 C 単位)。照会を処理するときにセッションが完全にアクティブになっていない場合は、このフィールドにゼロが戻されます。

**session_detail.session_timeout**
セッションと関連 U するタイムアウトを X 定します。これは、以下の値から得られます。

ローカル LU と関連する LU6.2 タイムアウト
リモート LU と関連する LU6.2 タイムアウト
モード・タイムアウト
グローバル・タイムアウト
限定 q 源タイムアウト (このセッションが限定 q 源リンクで実行されている場合)

**session_detail.plu_slu_comp_lvl**
PLU から SLU に送信されるデータの圧縮レベルを X 定します。

**AP_NONE**
圧縮は H 用されません。

**AP_RLE_COMPRESSION**
RLE 圧縮が H 用されます。

**AP_LZ9_COMPRESSION**
このノードは、LZ9 圧縮をサポートすることができます。

**AP_LZ10_COMPRESSION**
このノードは、LZ10 圧縮をサポートすることができます。

**AP_LZ12_COMPRESSION**
このノードは、LZ12 圧縮をサポートすることができます。

**session_detail.slu_plu_comp_lvl**
SLU から PLU に送信されるデータの圧縮レベルを X 定します。

**AP_NONE**
圧縮は H 用されません。

**AP_RLE_COMPRESSION**
RLE 圧縮が H 用されます。

**AP_LZ9_COMPRESSION**
このノードは、LZ9 圧縮をサポートすることができます。

**AP_LZ10_COMPRESSION**
このノードは、LZ10 圧縮をサポートすることができます。

**AP_LZ12_COMPRESSION**
このノードは、LZ12 圧縮をサポートすることができます。

**session_detail.rscv_len**

**session_detail** 構造体に追加された RSCV の長さ。 (何もU加されなかった場合、長さはゼロになります。) RSCV は、4 バイト境&の終わりまで埋め込まれます。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_SESSION_ID

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_LIST_OPTION

ノードが+ Oされないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_SIGNED_ON_LIST

QUERY_SIGNED_ON_LIST は、特定の LU にサインオンしているユーザーに関する情報を検索します。

ローカル LU は、 **lu_name** または **lu_alias** によってX定されます。 **Buf_ptr, buf_size, total_buf_size, num_entries, total_num_entries** と **overlay_size** は、QUERY verb として a 通の意味をもっています。

項目は、SIGNED_ON_LIST_ENTRY 構造体のリストとして戻されます。このリストは、**buf_ptr** によってポイントされるか、あるいは、**buf_ptr** がヌルの場合は、QUERY_SIGNED_ON_LIST VCB にU加されます。このリストは、まず **plu_alias/fqplu_name** 順に配列され、次に **user_id** 順、その次に **profile** 順に配列されます。 **plu_alias** をX定すると、 **fqplu_name** は無 k されます。

**list_options** オプションには、値 AP_FIRST_IN_LIST、AP_LIST_FROM_NEXT、または AP_LIST_INCLUSIVE を入れることができます。 **list_options** が AP_FIRST_IN_LIST であれば、 **plu_alias, fqplu_name, user_id** および **profile** は無 k されます。 **list** は、どのリストから項目を戻すか、どのリストを AP_SIGNED_ON_TO_LIST にする , 要があるかをX定します。

## VCB 構造体

```
typedef struct query_signed_on_list
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   *buf_ptr;           /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  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   user_id[10];        /* User ID                       */
        unsigned char   profile[10];        /* Profile                       */
        unsigned char   list;               /* Signed-on list type           */
} QUERY_SIGNED_ON_LIST;

typedef struct signed_on_list_entry
{
        unsigned short  overlay_size;       /* size of this entry            */
        unsigned char   plu_alias[8];       /* partner LU alias              */
        unsigned char   user_id[10];        /* fully qualified partner       */
        unsigned char   profile[10];        /* profile                       */
} SIGNED_ON_LIST_ENTRY;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_SIGNED_ON_LIST

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

これは、リスト情報に何を戻すかを示します。

**lu_name** (または、**lu_name** がすべてゼロに設定されている場合は **lu_alias**)、 **pu_alias** (または、**plu_alias** がすべてゼロに設定されている場合は **fqplu_name**)、 **user_id**、および **profile** を組み合わせたX定 (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

**pu_alias** および **fqplu_name** フィールドは無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から + O されます。

**lu_name**

LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。このフィールドをすべてゼロに設定すると、 **lu_alias** フィールドが索引の判別にH用されます。

**lu_alias**

ローカル定義 LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。このフィールドは、**lu_name** フィールドをすべてゼロに設定した場合にのみ有効です。この場合、8 バイトすべてが有効であるため、8 バイトすべてを設定する , 要があります。 **lu_name** および **lu_alias** の両フィールドをすべてゼロに設定すると、制御点 (デフォルト LU) と関連する LU がH用されます。

**plu_alias**

パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの

文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。このフィールドをすべてゼロに設定すると、 **fqplu_name** フィールドが索引値の判別に H 用されます。

**fqplu_name**
パートナー LU の 17 バイトの完全修飾ネットワーク名。この名前は、 1 つの EBCDIC ドットで連結された 2 つのタイプ A の EBCDIC 文 z 列からなり、右側に EBCDIC スペースが埋め込まれています。(それぞれの名前は、埋め込みスペースのない最大 8 バイトの長さになります。)

**user_id**
ユーザー ID。このフィールドは、10 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) に設定し、右側に EBCDIC スペースを埋め込まなければなりません。このフィールドを設定すると、このモードと関連するセッションのみが戻されます。 **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**profile**
10 バイトの英数 z EBCDIC 文 z 列。現在、「プログラム」は、ブランク・プロファイル (10 EBCDIC スペース) のみをサポートしています。 **list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

**list** サインオンされた q 源タイプ。これは、AP_SIGNED_ON_TO_LIST に設定しなければなりません。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**signed_on_list_entry.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**signed_on_list_entry.plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効です。

**signed_on_list_entry.user_id**
ユーザー ID。これは、10 バイト英数 z のタイプ A の EBCDIC 文 z 列 (文 z で O まる) で、右側に EBCDIC スペースが埋め込まれています。

**signed_on_list_entry.profile**
10 バイトの英数 z EBCDIC 文 z 列。現在、「プログラム」は、ブランク・プロファイル (10 EBCDIC スペース) のみをサポートしています。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_ALIAS

AP_INVALID_LU_NAME
AP_INVALID_PLU_NAME
AP_INVALID_USERID
AP_INVALID_PROFILE
AP_INVALID_LIST
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ するために verb が実行されない場合、「プログラム」は次のパラメーターが戻されます。

**primary_rc**
AP_NODE_STOPPING

# QUERY_STATISTICS

QUERY_STATISTICS は、リンク・ステーションとポートの統計を照会します。パーソナル・コミュニケーションズまたは Communications Serverは、この照会を DLC に直接渡します。統計の形式は、DLC インプリメンテーションによって異なります。

## VCB 構造体

```
typedef struct query_statistics
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   name[8];            /* LS 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;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_STATISTICS

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**name** リンク・ステーションまたはポートとして定義された名前 (**stats_type** パラメーターの設定によって異なります)。これは、ローカル＝示可　化　は家　億

**AP_ADMIN_TBL**
管理情報が戻されることをX定します。

**AP_OPER_TBL**
操作情報が戻されることをX定します。 F カテゴリーごとに戻される情報の形式は、 DLC インプリメンテーションによって異なります。

**reset_stats**
統計をリセットするかどうかをX定します (AP_YES または AP_NO)。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**dlc_type**
DLC のタイプ。このフィールドの値は、DLC インプリメンテーションに固有なものです。値は、以下のとおりです。

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

**statistics**
リンク・ステーションまたはポートの現在の統計。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LINK_NAME

AP_INVALID_PORT_NAME
AP_INVALID_STATS_TYPE
AP_INVALID_TABLE_TYPE

状態エラーのために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_LINK_DEACTIVATED

AP_PORT_DEACTIVATED

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_TP

QUERY_TP は、現在ローカル LU によってH用されているトランザクション・プログラムに関する情報を戻します。

この情報はリストとして戻されます。特定のトランザクション・プログラムに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、**tp_name** フィールドを設定する , 要があります。 **list_options** フィールドを AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。**lu_name** または **lu_alias** フィールドは、常に設定されていなければならないことに注意してください。**lu_name** フィールドは、非ゼロの場合、**lu_alias** フィールドに優先してH用されます。リスト形式のH用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

名前の長さが同じ場合は、このリストは、EBCDIC 辞書配列をH用して **tp_name** 順に配列されます。この verb は、ローカル LU が TP をH用しOめるときに決定された情報を戻します。 QUERY_TP_DEFINITION verb は、定義情報のみを戻します。

## VCB 構造体

```
typedef struct query_tp
{
        unsigned short  opcode;              /* Verb operation code            */
        unsigned char   attributes;          /* verb attributes                */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* Primary return code            */
        unsigned long   secondary_rc;        /* Secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   tp_name[64];         /* TP name                        */
        unsigned char   description[RD_LEN]; /* resource description           */
        unsigned short  instance_limit;      /* max instance count             */
        unsigned short  instance_count;      /* current instance count         */
        unsigned short  locally_started_count;
                                             /* locally started instance       */
                                             /* count                          */
        unsigned short  remotely_started_count;
                                             /* remotely started instance      */
                                             /* count                          */
        unsigned char   reserva[20];         /* reserved                       */
} TP_DATA;

typedef struct tp_spec_data
{
        unsigned char pathname[256];         /* path and TP name               */
        unsigned char parameters[64];        /* parameters for TP              */
        unsigned char queued;                /* queued TP (AP_YES)             */
```

```
        unsigned char load_type;            /* type of load-DETACHED/CONSOLE  */
        unsigned char dynamic_load;         /* dynamic loading of TP enabled  */
        unsigned char reserved[5];          /* max size is 120 bytes          */
} TP_SPEC_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_TP

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

これは、リスト情報に何を戻すかを示します。つまり、**lu_name** (または、**lu_name** がすべてゼロに設定されている場合は、**lu_alias**)、および **tp_name** を組み合わせたX定 (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**

索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**

戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**

戻りリストは、索引値によって示された項目から + O されます。

**lu_name**
LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。このフィールドをすべてゼロに設定すると、 **lu_alias** フィールドが索引の判別に H 用されます。

**lu_alias**
ローカル定義 LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。このフィールドは、**lu_name** フィールドをすべてゼロに設定した場合にのみ有効です。この場合、8 バイトすべてが有効であるため、8 バイトすべてを設定する , 要があります。 **lu_name** と **lu_alias** の両方をすべてゼロに設定すると、制御点 (デフォルトの LU) に関連する LU が H 用されます。

**tp_name**
トランザクション・プログラム名。これは、64 バイトの文 z 列で、右側にスペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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 , 要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**tp_data.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**tp_data.tp_name**
トランザクション・プログラム名。これは、64 バイトの文 z 列で、右側にスペースが埋め込まれています。

**tp_data.instance.description**
q 源の説明 (DEFINE_TP で X 定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**tp_data.instance_limit**
X 定されたトランザクション・プログラムで同 ~ にアクティブになっているインスタンスの最大数。

**tp_data.instance_count**
X定されたトランザクション・プログラムのインスタンスのうち、現在アクティブになっているものの数。

**tp_data.locally_started_count**
X定されたトランザクション・プログラムのインスタンスのうち、ローカルに + O された (TP_STARTED verb を発行するトランザクション・プログラムによって) ものの数。

**tp_data.remotely_started_count**
X定されたトランザクション・プログラムのインスタンスのうち、リモートに + O された (受信された接続要求によって) ものの数。

**tp_chars.tp_data.pathname**
パスおよびトランザクション・プログラム名をX定します。

**tp_chars.tp_data.parameters**
トランザクション・プログラムのパラメーターをX定します。

**tp_chars.tp_data.queued**
トランザクション・プログラムを待ち行列化するかどうかをX定します。

**tp_chars.tp_data.load_type**
トランザクション・プログラムをロードする方法をX定します。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TP_NAME

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_LIST_OPTION

ノードが + O されないためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_TP_DEFINITION

QUERY_TP_DEFINITION は、 DEFINE_TP verb ですでに渡されている情報、およびパーソナル・コミュニケーションズまたは Communications Serverによって定義されたトランザクション・プログラムに関する情報の両方を戻します。

この情報は、要約情報または詳細情報のいずれかの形式のリストとして戻されます。特定のトランザクション・プログラムに関する情報またはいくつかの 『固まり』 に分けられたリスト情報を入手するには、 **tp_name** フィールドを設定する , 要があります。

そうしないと ( **list_options** フィールドが AP_FIRST_IN_LIST に設定されている場合)、このフィールドは無 k されます。リスト形式の H 用方法に関する 2 考情報については、 10ページの『ノードの照会』 を 2 照してください。

このリストは、EBCDIC 辞書配列を H 用して、**tp_name** 順に配列されます。AP_LIST_FROM_NEXT を選択すると、戻りリストは、定義された順序に従って次の項目から + O されます (X定された項目が存在するかしないかに関係なく)。

この verb は、定義情報のみを戻します。 QUERY_TP verb は、ローカル LU がトランザクション・プログラムを H 用し O めるときに決定された情報を戻します。

## VCB 構造体

```
typedef struct query_tp_definition
{
        unsigned short  opcode;              /* Verb operation code            */
        unsigned char   attributes;          /* verb attributes                */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* Primary return code            */
        unsigned long   secondary_rc;        /* Secondary return code          */
        unsigned char   *buf_ptr;            /* pointer to buffer              */
        unsigned long   buf_size;            /* buffer size                    */
        unsigned long   total_buf_size;      /* total buffer size required     */
        unsigned short  num_entries;         /* number of entries              */
        unsigned short  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
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   tp_name[64];         /* TP name                        */
        unsigned char   description[RD_LEN]; /* resource description           */
} TP_DEF_SUMMARY;

typedef struct tp_def_detail
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   tp_name[64];         /* TP name                        */
        TP_CHARS        tp_chars;            /* TP characteristics             */
} TP_DEF_DETAIL;

typedef struct tp_chars
{
        unsigned char   description[RD_LEN]; /* resource description           */
        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   duplex_support;      /* duplex supported              */
        unsigned char   reserv3[9];          /* reserved                      */
        unsigned short  tp__instance_limit;  /* limit on currently active TP  */
                                             /* instances                     */
        unsigned short  incoming_alloc_timeout;
                                             /* incoming allocation timeout   */
        unsigned short  rcv_alloc_timeout;   /* receive allocation timeout    */
        unsigned short  tp_data_len;         /* TP data length                */
        TP_SPEC_DATA    tp_data;             /* TP data                       */
} TP_CHARS;

typedef struct tp_spec_data
{
        unsigned char pathname[256];         /* path and TP name              */
        unsigned char parameters[64];        /* parameters for TP             */
        unsigned char queued;                /* queued TP (AP_YES)            */
        unsigned char load_type;             /* type of load-DETACHED/CONSOLE */
        unsigned char dynamic_load;          /* dynamic loading of TP enabled */
        unsigned char reserved[5];           /* max size is 120 bytes         */
} TP_SPEC_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションでは、以下のパラメーターが提供されます。

**opcode**

AP_QUERY_TP_DEFINITION

**attributes**

この verb の属性。このフィールドはビット・フィールドです。最初のビットには、定義する q 源の可 k 性が含まれており、以下のいずれかと対応しています。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記リストの VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報を書き込むことができるバッファーをX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。その場合には、**buf_ptr** をヌルに設定しなければなりません。

**buf_size**

提供されたバッファーのサイズ。戻されたデータがこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数がこの値を超えることはありません。ゼロの値は限度がないことを意味しています。

**list_options**

この値は、リスト情報に何を戻すかを示します。

**AP_SUMMARY**
要約情報のみを戻します。

**AP_DETAIL**
詳細情報を戻します。

X定された **tp_name** (以下のパラメーターを 2 照) は、戻された実際の情報の + O点をX定するためにH用する索引値を示しています。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から + O されます。

**AP_LIST_FROM_NEXT**
戻りリストは、提供された索引値によってX定された項目の次のリスト内項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって示された項目から + O されます。

**tp_name**
定義済みのトランザクション・プログラムの名前。これは、64 バイトの文 z 列で、右側にスペースが埋め込まれています。**list_options** を AP_FIRST_IN_LIST に設定すると、このフィールドは無 k されます。

## 戻りＱｉａー？ー

この verb が正常に実行されると、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_OK

**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，要なバッファーのサイズを示す戻り値。この値は、**buf_size** の値より大きくすることができます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の数より大きくすることができます。

**tp_def_summary.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**tp_def_summary.tp_name**
定義済みのトランザクション・プログラム名。これは、64 バイトの文 z 列で、右側にスペースが埋め込まれています。

**tp_def_summary.description**
q 源の説明 (DEFINE_TP でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**tp_def_detail.overlay_size**
この項目内のバイトの数。つまり、戻された次の項目に対するオフセット (その項目が存在する場合)。

**tp_def_detail.tp_name**
定義済みのトランザクション・プログラム名。これは、64 バイトの文 z 列で、右側にスペースが埋め込まれています。

**tp_def_detail.tp_chars.description**
q 源の説明 (DEFINE_TP でX定)。これは、ローカル = 示可能文 z セットの 16 バイトの文 z 列です。16 バイトすべてが有効です。

**tp_def_detail.tp_chars.conv_type**
トランザクション・プログラムによってサポートされている会話のタイプをX定します。

AP_BASIC
AP_MAPPED
AP_EITHER

**tp_def_detail.tp_chars.security_rqd**
トランザクション・プログラムを + Oするために会話機密情報が , 要であるかどうかを示します (AP_NO または AP_YES)。

**tp_def_detail.tp_chars.sync_level**
トランザクション・プログラムによってサポートされている同期レベルをX定します。

**AP_NONE**
トランザクション・プログラムは、None の同期レベルをサポートします。

**AP_CONFIRM_SYNC_LEVEL**
トランザクション・プログラムは、Confirm の同期レベルをサポートします。

**AP_EITHER**
トランザクション・プログラムは、None または Confirm の同期レベルをサポートします。

**AP_SYNCPT_REQUIRED**
トランザクション・プログラムは、Sync-point の同期レベルをサポートします。

**AP_SYNCPT_NEGOTIABLE**
トランザクション・プログラムは、None、Confirm、または Sync-point の同期レベルをサポートします。

**tp_def_detail.tp_chars.dynamic_load**
トランザクション・プログラムを動的にロードできるかどうかをX定します (AP_YES または AP_NO)。

**tp_def_detail.tp_chars.enabled**
トランザクション・プログラムを正常に接続できるかどうかをX定します (AP_YES または AP_NO)。デフォルトは AP_NO です。

**tp_def_detail.tp_chars.pip_allowed**
トランザクション・プログラムが、プログラム初期設定 (PIP) パラメーターを受信できるかどうかをX定します (AP_YES または AP_NO)。

**tp_def_detail.tp_chars.duplex_support**
トランザクション・プログラムが全二重方式であるのか半二重方式であるのかを示します。

**AP_FULL_DUPLEX**
トランザクション・プログラムが全二重であることをX定します。

**AP_HALF_DUPLEX**
トランザクション・プログラムが半二重であることをX定します。

**AP_EITHER_DUPLEX**
トランザクション・プログラムを半二重または全二重にすることができることをX定します。

**tp_def_detail.tp_chars.tp_instance_limit**
同~にアクティブ状態にしておくことができるトランザクション・プログラム・インスタンスの数の限度。

**tp_def_detail.tp_chars.incoming_alloc_timeout**
着信接続が待ち行列で RECEIVE_ALLOCATE を待機しているC数をX定します。ゼロの値はタイムアウトがないことを示し、無期限の保留状態になります。

**tp_def_detail.tp_chars.rcv_alloc_timeout**
RECEIVE_ALLOCATE verb が待ち行列で接続を待機しているC数をX定します。ゼロの値はタイムアウトがないことを示し、無期限の保留状態になります。

**tp_def_detail.tp_chars.tp_data_len**
インプリメンテーション依存のトランザクション・プログラム・データの長さ。

**tp_def_detail.tp_chars.tp_data**
DYNAMIC_LOAD_INDICATION で未変更のまま渡されるインプリメンテーション依存のトランザクション・プログラム・データ。

パラメーター・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TP_NAME

AP_INVALID_LIST_OPTION

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのためにこの verb が実行されなかった場合、この「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# 第7章 ；ーＵ・９Ｈ＂ verb

この章ではネットワーク・ノードで発行される verb を説明します。

# SAFE_STORE_TOPOLOGY

SAFE_STORE_TOPOLOGY はネットワーク・ノードでのみH用され、ノードが再O動される場合に後でアクセス可能なトポロジー情報を安全に保管します。 **restore** フラグは情報が保管 (AP_NO) されるか、またはアクセス (AP_YES) されるかをX示するためにH用されます。

ストア・ノード情報はフォーマットされたリストとして戻されます。特定のネットワーク・ノードについての情報を得たり、いくつかの 『chunks』 のリスト情報を得るには、**index** フィールドがセットされる , 要があります。

そうでない場合（ **list_options** フィールドに AP_FIRST_IN_LIST を設定する場合）、このフィールドは無 k されます。リスト形式のH用方法に関する背景知識については10ページの『ノードの照会』を 2 照してください。

このリストは、**index_node_name** に基づいて配列されます。まず名前の長さ順に配列され、名前の長さが同じ場合には、ASCII の辞書配列の順番になります（IBM の 6611 APPN MIB 配列に準拠)。次に、リストは **index_node_type** にUいて数 z 順に配列されます。 TG が保管または | 元される場合には、配列は **index.tg_dest_node_name** にUいて (MIB 配列)、次に **index.tg_dest_node_type** にUいて (数 z 順) 、そして 3 番目に **index.tg_number** にUいて (数 z 順) です。

SAFE_STORE_TOPOLOGY verb は SFS_ADJACENT_NN、SFS_NN_TOPOLOGY_NODE および SFS_NN_TOPOLOGY_TG verb に取って代わります。これは、照会オーバーレーとの相互変換をするのではなくて、制御ベクトルがトポロジーに = 示される過程でそれをH用してトポロジー情報を保管します。未知の制御ベクトルは保管および | 元がされ、またチェックサムが破壊データのトポロジーへの混入を防ぐために提供されます。

## VCB 構造体

```
typedef struct safe_store_topology
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   buf_ptr;             /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
                                             /* to hold all information       */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  total_num_entries;   /* total number of entries       */
        unsigned char   list_options;        /* listing options               */
        unsigned char   restore;             /* store or restore;             */
        unsigned char   resource_types;      /* resource types (nodes, TGs...) */
        RESOURCE_INDEX  index;               /* resource index                */
        unsigned long   frsn;                /* flow-reduction sequence       */
                                             /* number                        */
        unsigned char   reserv3[16];         /* reserved                      */
      } SAFE_STORE_TOPOLOGY;

typedef struct resource_index
{
        unsigned char   node_name[17];       /* FQ node name                  */
        unsigned char   node_type;           /* node type                     */
```

```
        unsigned char   tg_dest_node_name[17];
                                            /* FQ name of TG destination node */
        unsigned char   tg_dest_node_type;  /* TG destination node type       */
        unsigned char   tg_number;          /* TG number                      */
        unsigned char   reserv1[3];         /* reserved                       */
} RESOURCE_INDEX;

typedef struct safe_store_data
{
        unsigned short  overlay_size;       /* overalllength of safe         */
                                            /* store data                    */
        unsigned short  sub_overlay_size;   /* offset to first appended      */
                                            /* resource                      */
        RESOURCE_INDEX  index;              /* index of appended resource    */
        unsigned char   checksum[16];       /* reserved                      */
} RESOURCE_INDEX;

typedef struct safe_store_node_data
{
        unsigned short  overlay_size;        /* overalllength of safe          */
                                             /* store data                     */
        unsigned short  sub_overlay_size;    /* offset to first appended       */
        unsigned char   adjacent;            /* is this NNCP and adjacent      */
                                             /* NNCP?                          */
        unsigned char   reserv1;             /* reserved                       */
        unsigned long   last_frsn_sent;      /* last flow reduction sequence   */
                                             /* num sent (if node is adjacent) */
                                             /* resource                       */
        unsigned long   last_frsn_rcvd;      /* last flow reduction sequence   */
                                             /* num rcvd (if node is adjacent) */
        unsigned long   frsn;                /* flow reduction sequence number */
        unsigned short  days_left            /* days left in database          */
       RESOURCE_INDEX  index;                /* index of appended resource     */
} SAFE_STORE_NODE_DATA;

typedef struct safe_store_tg_data
{
        unsigned short  overlay_size;        /* overalllength of safe          */
                                             /* store data                     */
        unsigned short  sub_overlay_size;    /* offset to first appended       */
                                             /* resource                       */
        unsigned long   frsn;                /* flow reduction sequence number */
        unsigned short  days_left            /* days left in database          */
        unsigned short  vector_len;          /* length of appended vector(s)   */
} SAFE_STORE_TG_DATA;
```

## 指定Ｑｉａー？ー

### restore = AP_NO のときの指定パラメーター

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_SAFE_STORE_TOPOLOGY

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定しなければなりません。

## SAFE_STORE_TOPOLOGY

**buf_size**
提供されるバッファーのサイズ。戻りデータはこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**list_options**
ここでは、リスト情報として何を戻すかを X 定します。 X 定された **resource_types** と **index** ( 下のパラメーターを 2 照）は、戻される実情報の + O 点を示すために H 用される索引値を = します。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から O まります。

**AP_LIST_FROM_NEXT**
戻りリストは、システムに提供された索引値が示す項目の次の項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって X 定される項目から O まります。

**restore**
フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**resource_types**
このビット・フィールドは保管されるトポロジー・データを制御します。以下の値のどの組み合わせもこのフィールドでビット単位で互いに OR 結合されます。

**AP_SFS_NODES**
トポロジー・ノードを保管します。

**AP_SFS_ADJ_NODES**
隣接ノードを保管します。

**AP_SFS_TGS**
TG を保管します。

**注:** これらの 3 つのフラグの内少なくとも 1 つはセットされる , 要があります。隣接ノードとトポロジー・ノードは APPN 内で別々のエンティティーです、したがって最初の 2 つのフラグはどんな組み合わせも可能です。

**index.node_name**
ネットワーク・トポロジー・データベースから得られたネットワーク修飾ノード名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。 EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。) このフィールドは、APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされて

いると無 k されます。このフィールドは、AP_SFS_NODES も AP_SFS_ADJ_NODES もともに **resource_types** の中にセットされていない場合にも無 k されます。

**index.node_type**
ノードのタイプ。以下のいずれかに設定されます。

AP_NETWORK_NODE
AP_VRN
AP_LEARN_NODE

**node_type** が T 明の場合、AP_LEARN_NODE を X 定する , 要があります。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。このフィールドは、AP_SFS_NODES も AP_SFS_ADJ_ADJ もともに **resource_types** の中にセットされていない場合にも無 k されます。

**index.tg_dest_node_name**
TG の完全修飾宛先ノード名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。) このフィールドは、APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。このフィールドは、AP_SFS_NODES も AP_SFS_ADJ_NODES もともに **resource_types** の中にセットされていない場合にも無 k されます。

**index.tg_dest_node_type**
この TG の宛先ノードのタイプ。以下のいずれかに設定されます。

AP_NETWORK_NODE
AP_VRN

**tg_dest_node_type** が T 明の場合、AP_LEARN_NODE を X 定する , 要があります。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。このフィールドは、AP_SFS_TGS が **resource_types** の中にセットされていない場合にも無 k されます。

**index.tg_number**
TG と関連した番号。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。このフィールドは、AP_SFS_TGS が **resource_types** の中にセットされていない場合にも無 k されます。

**frsn** フロー縮約シーケンス番号 (frsn)。これが非ゼロの場合、この値以上の FRSN を } つトポロジー・リソースのみが戻されます。

**safe_store_data.overlay_size**
この記入項目の長さ。埋め込みも含みます。これは次の SAFE_STORE_DATA オーバーレーに対するオフセット (ある場合)。

**safe_store_data.sub_overlay_size**
この記入項目の長さ。埋め込みも含みます。これは U 加された

SAFE_STORE_DATA または SAFE_STORE_TG_DATA に対するオフセット。このフィールドはU加データをアクセスするときには常にH用する，要があります。

**safe_store_data.index**
この記入項目の索引。この構造体は後続の記入項目をリストするために後続の SAFE_STORE_TOPOLOGY verb のシステムに提供できます。**dest_tg_name** がすべて 2 進ゼロにセットされている場合には、SAFE_STORE_NODE_DATA オーバーレーが後に続きます。そうでない場合には、SAFE_STORE_TG_DATA オーバーレーが後に続きます。

**safe_store_data.checksum**
U加されたオーバーレーとベクトルの 128 ビット・チェックサム。このチェックサムと次のデータが破壊される場合には、高いN立で破壊が検出されて、verb がリジェクトされます。

**safe_store_node_data.overlay_size**
この記入項目の長さ。埋め込みも含みます。これはU加された SAFE_STORE_DATA または SAFE_STORE_TG_DATA に対するオフセット。

**safe_store_node_data.sub_overlay_size**
この記入項目の長さ。埋め込みも含みます。これはU加された SAFE_STORE_DATA または SAFE_STORE_TG_DATA に対するオフセット。このフィールドはU加ベクトルをアクセスするためには常にH用する，要があります。

**safe_store_node_data.adjacent**
AP_YES または AP_NO 。AP_YES の場合にはこの記入項目は隣接ネットワーク・ノードに一致します。

**safe_store_node_data.last_frsn_sent**
**adjacent** が AP_YES にセットされている場合には、このフィールドは隣接ネットワーク・ノードに送信された最後の FRSN を保 } します。そうでない場合には、このフィールドはゼロにセットされます。

**safe_store_node_data.last_frsn_rcvd**
**adjacent** が AP_YES にセットされている場合には、このフィールドは隣接ネットワーク・ノードに送信された最後の FRSN を保 } します。そうでない場合には、このフィールドはゼロにセットされます。

**safe_store_node_data.frsn**
トポロジー・リソースのフロー縮約シーケンス番号 (このノードがトポロジーに現れる場合)。そうでない場合には、このフィールドはゼロにセットされます。

**safe_store_node_data.days_left**
このノードの存在がN認できない場合に、ノードが除去される前にトポロジー・データベースにD留する日数。ゼロは限&がないことを意味します。

**safe_store_node_data.vector_len**
U加ベクトルの長さ。ゼロは、ベクトルがU加されないことを意味します。

**safe_store_tg_data.overlay_size**
この記入項目の長さ。埋め込みも含みます。これはU加された SAFE_STORE_DATA または SAFE_STORE_TG_DATA に対するオフセット。

**safe_store_tg_data.sub_overlay_size**
この記入項目の長さ。埋め込みも含みます。これはU加されたベクトルに対するオフセット、(ある場合)。このフィールドはU加ベクトルをアクセスするために常にH用する，要があります。

**safe_store_tg_data.frsn**
この TG のフロー縮約シーケンス番号。

**safe_store_tg_data.days_left**
この TG の存在がN認できない場合に、TG が除去される前にトポロジー・データベースにD留する日数。ゼロは限&がないことを意味します。

**safe_store_node_data.vector_len**
U加ベクトルの長さ。ゼロは、ベクトルがU加されないことを意味します。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**buf_size**
バッファーに戻った情報の長さ。

**total_buf_size**
要求されたすべてのリスト情報を戻すために，要になるバッファー・サイズを示す戻り値。これは **buf_size** の値より大きくできます。

**total_num_entries**
戻すことができたはずの項目の合計数。これは **num_entries** の数より大きくできます。

**num_entries**
実際に戻された項目の数。

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LIST_OPTION

AP_INVALID_NODE
AP_INVALID_RESOURCE_TYPES
AP_INVALID_TG

## 指定Ｑｉａー？ー

**restore = AP_YES のときの指定パラメーター**

アプリケーションは以下のパラメーターをX定します。

### SAFE_STORE_TOPOLOGY

**opcode**

AP_SAFE_STORE_TOPOLOGY

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定し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。戻りデータはこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**restore**

フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**resource_name**

ネットワークの完全修飾リソース名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。（それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。）このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**resource_type**

このビット・フィールドは保管されるトポロジー・データを制御します。以下の値のどの組み合わせもこのフィールドでビット単位で互いに OR 結合されます。

**AP_SFS_NODES**

トポロジー・ノードを | 元します。

**AP_SFS_ADJ_NODES**

隣接ノードを | 元します。

**AP_SFS_TGS**

TG を | 元します。

**注:** これらの 3 つのフラグの内少なくとも 1 つはセットされる , 要があります。隣接ノードとトポロジー・ノードは APPN 内で別々のエンティティーです、したがって最初の 2 つのフラグはどんな組み合わせも可能です。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_CHECKSUM_FAILED

AP_DATA_CORRUPT
AP_INVALID_RESOURCE_TYPES

関係のある START_NODE パラメーターがセットされなかったために verb が実行されない場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ネットワーク・ノード・サポートをH用してシステムが構築されていないために verb が実行されない場合、「プログラム」は次のパラメーターが戻されます。

**primary_rc**
AP_INVALID_VERB

ノードがまだ+ Oされていないために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# SFS_ADJACENT_NN

**注:** この verb は SAFE_STORE_TOPOLOGY で取り代えられ、「プログラム」の直前バージョンとの互換性のためだけにリテインされます。

SFS_ADJACENT_NN は、ノードが再O動される場合に後でアクセス可能なトポロジー情報を安全に保管するためにH用されます。**restore** フラグは情報が保管 (AP_NO) されるか、またはアクセス (AP_YES) されるかをX示するためにH用されます。

**restore** フラグが AP_NO にセットされているときには、SFS_ADJACENT_NN は隣接ネットワーク・ノードについての情報を戻します。 (すなわち、その CP-CP セッションがアクティブであるネットワーク・ノードがアクティブになっているか、またはいつかアクティブになったことがあります。)

SFS 情報はフォーマットされたリストとして戻されます。特定のネットワーク・ノードについての情報、またはいくつかの『固まり』に分かれたリスト情報を得る場合は、 **adj_nncp_name** フィールドを設定する , 要があります。

そうでない場合（ **list_options** フィールドに AP_FIRST_IN_LIST を設定する場合）、このフィールドは無 k されます。リスト形式のH用方法に関する背景知識については10ページの『ノードの照会』を 2 照してください。

このリストは、**adj_nncp_name** に基づいて配列されます。まず名前の長さ順に配列され、名前の長さが同じ場合には、ASCII の辞書配列の順番になります（IBM の 6611 APPN MIB 配列に準拠）。AP_LIST_FROM_NEXT が選択される場合、リストは、定義された配列に従って（X定された項目があっても、なくても）次の項目からOまります。

## VCB 構造体

```
typedef struct sfs_adjacent_nn
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
                                             /* to hold all information       */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  total_num_entries;   /* total number of entries       */
        unsigned char   list_options;        /* listing options               */
        unsigned char   restore;             /* store or restore;             */
        unsigned char   adj_nncp_name[17];   /* CP name of adj Network Node   */
} SFS_ADJACENT_NN;

typedef struct adj_nncp_data
{
        unsigned short  overlay_size;         /* size of this entry           */
        unsigned char   adj_nncp_name[17];    /* CP name of adj Network Node  */
        unsigned char   cp_cp_sess_status;    /* CP-CP session status         */
        unsigned COUNTER out_of_seq_tdus;     /* out of sequence TDUs         */
        unsigned long   last_frsn_sent;       /* last FSRN sent               */
        unsigned long   last_frsn_rcvd;       /* last FRSN received           */
        unsigned char   reserva[20];          /* reserved                     */
} ADJ_NNCP_DATA;
```

## 指定Ｑｉａー？ー

### restore = AP_NO のときの指定パラメーター

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_SFS_ADJACENT_NN

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定し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。戻りデータはこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**list_options**
ここでは、リスト情報として何を戻すかをX定します。X定された **resource_types** と **index** ( 下のパラメーターを2照）は、戻される実情報の+O点を示すためにH用される索引値を=します。

**AP_FIRST_IN_LIST**
索引値は無kされ、戻りリストはリスト内の最初の項目からOまります。

**AP_LIST_FROM_NEXT**
戻りリストは、システムに提供された索引値が示す項目の次の項目から+Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によってX定される項目からOまります。

**restore**
フラグは情報が|元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**adj_nncp_name**
隣接ネットワーク・ノードの 17 バイト完全修飾 CP 名。この名前は、2 つのタイプ A EBCDIC 文z列を EBCDIC のドットによって区切り、右側の余白に EBCDIC のスペースを埋め込んだ形式で構成されます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。) このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無kされます。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**buf_size**
バッファーに戻った情報の長さ。

**total_buf_size**
要求されたすべてのリスト情報を戻すために，要になるバッファー・サイズを示す戻り値。これは **buf_size** の値より大きくできます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。これは **num_entries** の数より大きくできます。

**adj_nncp_data.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。

**adj_nncp_data.adj_nncp_name**
隣接ネットワーク・ノードの17バイト完全修飾 CP 名。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットによって区切り、右側の余白に EBCDIC スペースを埋め込んだ形式で X 定します。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

**adj_nncp_data.cp_cp_sess_status**
CP-CP セッションの状況 (AP_ACTIVE または AP_INACTIVE)。

**adj_nncp_data.out_of_seq_tdus**
ノードから受信した順 T 同の TDU の数。

**adj_nncp_data.last_frsn_sent**
このノードに送信された最終フロー縮約シーケンス番号。

**adj_nncp_data.last_frsn_rcvd**
このノードから受信された最終フロー縮約シーケンス番号。

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_OK

**secondary_rc**
AP_INVALID_ADJ_NNCP_NAME

AP_INVALID_LIST_OPTION

## 指定Ｑｉａー？ー

### restore = AP_YES のときの指定パラメーター

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_SFS_ADJACENT_NN

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定しなければなりません。

**num_entries**
実際に戻された項目の数。

**restore**
フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**adj_nncp_data.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。

**adj_nncp_data.adj_nncp_name**
隣接ネットワーク・ノードの17バイト完全修飾 CP 名。この名前は、2 つのタイプ A の EBCDIC 文z列を EBCDIC ドットによって区切り、右側の余白に EBCDIC スペースを埋め込んだ形式でX定します。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。)

**adj_nncp_data.cp_cp_sess_status**
**restore** が AP_YES に設定されていると、このフィールドは無k されます。

**adj_nncp_data.out_of_seq_tdus**
**restore** が AP_YES に設定されていると、このフィールドは無k されます。

**adj_nncp_data.last_frsn_sent**
このノードに送信された最終フロー縮約シーケンス番号。

**adj_nncp_data.last_frsn_rcvd**
このノードから受信された最終フロー縮約シーケンス番号。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

ノードが+Oされていないために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

関係する START_NODE パラメーターが送信されなかったために verb が実行されない場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ネットワーク・ノード・サポートをH用してシステムが構築されていないために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_INVALID_VERB

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# SFS_DIRECTORY

QUERY_DIRECTORY_ENTRY verb に加えて、ネットワーク・ノードのローカル・ディレクトリー・キャッシュが安全に保管されて、ノードがリスタートすれば後でアクセスできる SFS_DIRECTORY verb があります。 **restore** フラグは情報が保管 (AP_NO) されるか、またはアクセス (AP_YES) されるかをX示するためにH用されます。

**restore** フラグが AP_YES にセットされているときには、SFS_DIRECTORY はディレクトリー・データベースが **directory_entry_summary** オーバーレーをH用して再作成されることを可能にします。特定のネットワーク・ノードについての情報を得たり、いくつかの 『chunks』 のリスト情報を得るには, **resource_name** と **resource_type** フィールドがセットされている , 要があります。

そうでない場合（ **list_options** フィールドに AP_FIRST_IN_LIST を設定する場合）、このフィールドは無 k されます。リスト形式のH用方法に関する背景知識については10ページの『ノードの照会』を 2 照してください。

キャッシュ項目とその親に関するリソース情報は以下の順序で戻されます。

1 番目のネットワーク・ノード
- 第 1 LU (ネットワーク・ノードの)
- 第 2 LU (ネットワーク・ノードの)
- ...
- 第 n LU (ネットワーク・ノードの)

1 番目のエンド・ノード (このネットワーク・ノードがサーブする)
- 第 1 LU (エンド・ノード (1) の)
- 第 2 LU (エンド・ノード (1) の)
- ...
- 第 n LU (エンド・ノード (1) の)
- ...

n 番目のエンド・ノード (このネットワーク・ノードがサーブする)
- 第 1 LU (エンド・ノード (n) の)
- 第 2 LU (エンド・ノード (n) の)
- ...

2 番目のネットワーク・ノード
- ...

## VCB 構造体

```
typedef struct sfs_directory
{
        unsigned short  opcode;              /* verb operation code          */
        unsigned char   reserv2;             /* reserved                     */
        unsigned char   format;              /* format                       */
        unsigned short  primary_rc;          /* primary return code          */
        unsigned long   secondary_rc;        /* secondary return code        */
        unsigned char   *buf_ptr;            /* pointer to buffer            */
        unsigned long   buf_size;            /* buffer size                  */
        unsigned long   total_buf_size;      /* total buffer size required   */
        unsigned short  num_entries;         /* number of entries            */
        unsigned short  total_num_entries;   /* total number of entries      */
        unsigned char   list_options;        /* listing options              */
```

```
        unsigned char   restore;              /* store or restore flag         */
        unsigned char   resource_name[17];    /* network qualified res name    */
        unsigned char   reserv3;              /* reserved                      */
        unsigned short  resource_type;        /* Resource type                 */
} SFS_DIRECTORY;

typedef struct directory_entry_summary
{
        unsigned short  overlay_size;        /* size of this entry             */
        unsigned char   resource_name[17];   /* network qualified res name     */
        unsigned char   reserve1;            /* reserved                       */
        unsigned short  resource_type;       /* Resource type                  */
        unsigned short  real_owning_cp_type; /* real owning CP type            */
        unsigned char   real_owning_cp_name[17];
                                             /* real owning CP name            */
        unsigned char   description[RD_LEN]; /* resource description           */
} DIRECTORY_ENTRY_SUMMARY;
```

## 指定Ｑｉａー？ー

### restore = AP_NO のときの指定パラメーター

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_SFS_ADJACENT_NN

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定し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。戻りデータはこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**list_options**

ここでは、リスト情報として何を戻すかをX定します。X定された **resource_name** と **resource_type**（以下のパラメーターを2照）は、戻される実際の情報の+O点をX定するためにH用される索引値を=しています。

**AP_FIRST_IN_LIST**

索引値は無kされ、戻りリストはリスト内の最初の項目からOまります。

**AP_LIST_FROM_NEXT**

戻りリストは、システムに提供された索引値が示す項目の次の項目から+Oされます。

**restore**
フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**resource_name**
ネットワークの修飾リソース名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。それは EBCDIC ドットによって連結された 2 つのタイプ A EBCDIC の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく最大 8 バイトの長さです。) このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**resource_type**
q 源タイプ。以下のいずれかを 2 照してください。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**buf_size**
バッファーに戻った情報の長さ。

**total_buf_size**
要求されたすべてのリスト情報を戻すために , 要になるバッファー・サイズを示す戻り値。これは **buf_size** の値より大きくできます。

**total_num_entries**
戻すことができたはずの項目の合計数。これは **num_entries** の数より大きくできます。

**num_entries**
実際に戻された項目の数。

**directory_entry_summary.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。

**directory_entry_summary.resource_name**
ネットワークの修飾リソース名。この名前の長さは 17 バイトで、 EBCDIC スペースが右の余白に埋め込まれます。それは EBCDIC ドットによって連結された 2 つのタイプ A EBCDIC の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。) このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**directory_entry_summary.resource_type**
q 源タイプ。以下のいずれかを 2 照してください。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_RES_NAME

AP_INVALID_LIST_OPTION
AP_INVALID_RES_TYPE

**directory_entry_summary.real_owning_cp_type**
NN と BrNN のみ: 実際の所有する CP タイプ。これは以下のうちのどれかになります。

**AP_NONE**
実際の所有する CP は親 q 源です。

**AP_ENCP_RESOURCE**
実際の所有する CP は親 q 源でではなく EN です。

他のノード・タイプ : このフィールドは AP_NONE にセットされます。

**directory_entry_summary.real_owning_cp_name**
NN と BrNN のみ: 完全修飾の実際の所有する CP 名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。 2 つのタイプ A の EBCDIC 文 z 列を EBCDIC のドットで区切った形になります。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

実際の所有する CP が親である場合には、このフィールドは 2 進ゼロにセットされます。

実際の所有する CP が親でない場合には、このフィールドは実際の所有する CP の名前にセットされます。

リソースが BrNN のドメインにある EN によって所有される場合には、実際の所有する CP は BrNN の NNS のディレクトリーにある親ではありません。このケースでは、実際の所有する CP は EN です、しかし親は BrNN です。

他のノード・タイプ : このフィールドは 2 進ゼロにセットされます。

## 指定Ｑｉａー？ー

**resorte = AP_YES のときの指定パラメーター**

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_SFS_DIRECTORY

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定し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。

**restore**
フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**resource_name**
ネットワークの修飾リソース名。この名前の長さは 17 バイトで、 EBCDIC スペースが右の余白に埋め込まれます。それは EBCDIC ドットによって連結された 2 つのタイプ A EBCDIC の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）アプリケーションがディレクトリーの最初の 『chunk』 を | 元している場合には、これはすべてゼロにセットされます。そうでない場合には、アプリケーションはこれを直前の 『chunk』 の最後の項目の q 源名にセットすべきです。

**resource_type**
q 源タイプ。以下のいずれかを 2 照してください。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

アプリケーションがディレクトリーの最初の 『chunk』 を | 元している場合には、このフィールドはゼロにセットされるべきです。

**directory_entry_summary.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。これは **restore** が AP_NO にセットされているときに戻される **overlay_size** と同じである , 要があります。

**directory_entry_summary.resource_name**
ネットワークの修飾リソース名。この名前の長さは 17 バイトで、 EBCDIC スペースが右の余白に埋め込まれます。それは EBCDIC ドットによって連結された 2 つのタイプ A EBCDIC の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）

**directory_entry_summary.resource_type**
q 源タイプ。以下のいずれかを 2 照してください。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

**directory_entry_summary.real_owning_cp_type**

NN と BrNN のみ: 実際の所有する CP タイプ。これは以下のうちのどれかになります。

**AP_NONE**

実際の所有する CP は親 q 源です。

**AP_ENCP_RESOURCE**

実際の所有する CP は親 q 源でではなく、EN です。

他のノード・タイプ : このフィールドは AP_NONE にセットされます。

**directory_entry_summary.real_owning_cp_name**

NN と BrNN のみ: 完全修飾の実際の所有する CP 名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。 2 つのタイプ A の EBCDIC 文 z 列を EBCDIC のドットで区切った形になります。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

実際の所有する CP が親である場合には、このフィールドは 2 進ゼロにセットされます。

実際の所有する CP が親でない場合には、このフィールドは実際の所有する CP の名前にセットされます。

リソースが BrNN のドメインにある EN によって所有される場合には、実際の所有する CP は BrNN の NNS のディレクトリーにある親ではありません。このケースでは、実際の所有する CP は EN です、しかし親は BrNN です。

他のノード・タイプ : このフィールドは 2 進ゼロにセットされます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_RES_NAME

AP_INVALID_LIST_OPTION

関係する START_NODE パラメーターが送信されなかったために verb が実行されない場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_FUNCTION_NOT_SUPPORTED

ネットワーク・ノード・サポートをH用してシステムが構築されていないために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_INVALID_VERB

ノードが + O されていないために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# SFS_NN_TOPOLOGY_NODE

**注:** この verb は SAFE_STORE_TOPOLOGY で取り代えられ、「プログラム」の直前バージョンとの互換性のためだけにリテインされます。

それぞれのネットワーク・ノードは、ネットワーク内にネットワーク・ノード、VRN、およびネットワーク・ノード間の TG に関する情報を保 } するネットワーク・トポロジー・データベースを保 } します。 SFS_NN_TOPOLOGY_NODE は、ノードが再O動される場合に後でアクセス可能なトポロジー・データベース・ノード項目を安全に保管するためにH用されます。 **restore** フラグは情報が保管 (AP_NO) されるか、またはアクセス (AP_YES) されるかをX示するためにH用されます。

特定のネットワーク・ノードについての情報、またはいくつかの 『chunks』のリスト情報を得るには、**node_name** と **node_type** フィールドを設定する , 要があります。

そうでない場合（ **list_options** フィールドに AP_FIRST_IN_LIST を設定する場合）、このフィールドは無 k されます。リスト形式のH用方法に関する背景知識については10ページの『ノードの照会』を 2 照してください。

このリストは、**node_name**、 **node_name_type**、および **frsn** に基づいて配列されます。まず名前の長さ順に配列され、名前の長さが同じ場合には、ASCII の辞書配列の順番になります（IBM の 6611 APPN MIB 配列に準拠)。 **node_type** の配列は AP_NETWORK_NODE、次に AP_VRN の順です。**frsn** は数値順になります。

- AP_LIST_INCLUSIVE を選択すると、戻りリストはその名前の最初の有効なレコードから + O されます。
- AP_LIST_FROM_NEXT を選択すると、リストは、X定された名前の次の名前がついた最初の有効なレコードから + O されます。

**frsn** フィールドを非ゼロ値に設定すると、これより高いフロー縮約シーケンス番号 (FRSN) を } つデータベース項目のみが戻されることに注意してください。これは、まずノードの現行 FRSN を得ることによっていくつかの 『chunks』 内に一貫性のあるトポロジー・データベースを戻すことを可能にします。これは以下のように実行されます。

1. ノードの現行 FRSN を戻す QUERY_NODE を発行する。
2. , 要なだけ SFS_NN_TOPOLOGY_NODE（FRSN をゼロに設定して）を発行して、『chunks』 内のすべてのデータベース・エントリーをM得する。
3. QUERY_NODE を再発行して、新規作成 FRSN をステップ 1. で戻されたものと比 S する。
4. 2 つの FRSN が、データベース内で変更されたものと異なっている場合には、FRSN をステップ 1. で提供された FRSN より 1 だけ大きい値に設定して SFS_NN_TOPOLOGY_NODE を発行する。

## VCB 構造体

```
typedef struct sfs_nn_topology_node
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   reserv2;            /* reserved                     */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
```

```
        unsigned long   secondary_rc;       /* secondary return code          */
        unsigned char   *buf_ptr;           /* pointer to buffer              */
        unsigned long   buf_size;           /* buffer size                    */
        unsigned long   total_buf_size;     /* total buffer size required     */
        unsigned short  num_entries;        /* number of entries              */
        unsigned short  total_num_entries;  /* total number of entries        */
        unsigned char   list_options;       /* listing options                */
        unsigned char   restore;            /* store or restore;              */
        unsigned char   node_name[17];      /* network qualified              */
                                            /* node name                      */
        unsigned char   node_type;          /* node type                      */
        unsigned long   frsn;               /* flow-reduction sequence        */
                                            /* number                         */
} SFS_NN_TOPOLOGY_NODE;

typedef struct nn_topology_node_detail
{
        unsigned short  overlay_size;       /* size of this entry             */
        unsigned char   node_name[17];      /* network qualified              */
        unsigned char   node_type;          /* node type                      */
        unsigned short  days_left           /* days left in database          */
        unsigned long   frsn;               /* flow reduction sequence number */
        unsigned long   rsn;                /* resource sequence number       */
        unsigned char   rar;                /* route additional resistence    */
        unsigned char   status;             /* node status                    */
        unsigned char   function_support;   /* function support               */
        unsigned char   reserv2;            /* reserved                       */
        unsigned char   reserva[20];        /* reserved                       */
} NN_TOPOLOGY_NODE_DETAIL;
```

## 指定Ｑｉａー？ー

### restore = AP_NO のときの指定パラメーター

アプリケーションは以下のパラメーターをＸ定します。

**opcode**

AP_SFS_NN_TOPOLOGY_NODE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをＸ定するには、このフィールドをゼロに設定します。

**buf_ptr**

リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定し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。戻りデータはこのサイズを超えることはありません。

**num_entries**

戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**list_options**

ここでは、リスト情報として何を戻すかをＸ定します。Ｘ定された **node_name, node_types** と **frsn** (下のパラメーターを２照) は戻される実際の情報の＋Ｏ点をＸ定するためにＨ用される索引値を＝します。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から O まります。

**AP_LIST_FROM_NEXT**
戻りリストは、システムに提供された索引値が示す項目の次の項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって X 定される項目から O まります。

**restore**
フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**node_name**
ネットワーク・トポロジー・データベースから得られたネットワーク修飾ノード名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。 EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。) このフィールドは、APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**node_type**
ノードのタイプ。以下のいずれかに設定されます。

AP_NETWORK_NODE
AP_VRN

**node_type** が T 明の場合、AP_LEARN_NODE を X 定する , 要があります。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**frsn** フロー縮約シーケンス番号。これが非ゼロの場合、この値以上の FRSN を } つトポロジー・リソースのみが戻されます。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**buf_size**
バッファーに戻った情報の長さ。

**total_buf_size**
要求されたすべてのリスト情報を戻すために , 要になるバッファー・サイズを示す戻り値。これは **buf_size** の値より大きくできます。

**total_num_entries**
戻すことができたはずの項目の合計数。これは **num_entries** の数より大きくできます。

**num_entries**
実際に戻された項目の数。

**nn_topology_node_detail.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。

**nn_topology_node_detail.node_name**
ネットワーク・トポロジー・データベースから得られたネットワーク修飾ノード名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**nn_topology_node_detail.node_type**
ノードのタイプ。以下のいずれかになります。

AP_NETWORK_NODE
AP_VRN

**nn_topology_node_detail.days_left**
トポロジー・データベースからこのノード項目を削除するまでの日数。ローカル・ノード項目には、これはゼロに設定されます（この項目は削除されることはありません)。これはレコードが | 元されるときにはゼロにリセットされる , 要があります (たとえば、**restore** は AP_YES にセットされます)。

**nn_topology_node_detail.frsn**
フロー縮約シーケンス番号。これは q 源がローカル・ノードで更新された最終 ~ 刻を示します。

**nn_topology_node_detail.rsn**
q 源シーケンス番号。これは、この q 源を所有するネットワーク・ノードによって d り当てられます。

**nn_topology_node_detail.rar**
ネットワーク・ノードの経路追加レジスタンス。

**nn_topology_node_detail.status**
このフィールドはノードの状況を X 定し、AP_UNCONGESTED、または以下の内から 1 つまたは複数が互いに OR 結合されたものが可能です。

**AP_CONGESTED**
ISR セッションの数が START_NODE verb で X 定された **isr_sessions_upper_threshold** を越えています。

**AP_IRR_DEPLETED**
ISR セッションの数が START_NODE verb の **max_isr_sessions** パラメーターで X 定された最大値に達しました。

**AP_ERR_DEPLETED**
エンドポイント・セッションの数が X 定された最大数に到達しました。

**AP_QUIESCING**
タイプ AP_QUIENCE または AP_QUIENCE_ISR の STOP_NODE が発行されました。

**nn_topology_node_detail.function_support**
これはどの機能がサポートされるかをX定します。これは以下の内の 1 つまたは複数が可能です。

**AP_BORDER_NODE**
ボーダー・ノード・ファンクションがサポートされます。

**AP_CDS**
中央ディレクトリー・サーバーがサポートされます。

**AP_GATEWAY**
ノードはゲートウェイ・ノード (機能はいまだ構造定義されていない) です。

**AP_ISR**
このノードは中間セッション経路X定 (ISR) をサポートします。

**AP_HPR**
このノードは中間セッション経路X定 (ISR) をサポートします。

**AP_RTP_TOWER**
このノードは、HPR の RTP タワーをサポートします。

**AP_CONTROL_OVER_RTP_TOWER**
ノードは、RTP タワーを介した制御フローをサポートします。

**注:** AP_CONTROL_OVER_RTP_TOWER ノードは AP_HPR と AP_RTP_TOWER の両方の設定に対応します。

verb が正常に実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LIST_OPTION

AP_INVALID_NODE
AP_INVALID_LIST_OPTIONS

## 指定Ｑｉａー？ー

**restore = AP_YES のときの指定パラメーター**

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_SFS_NN_TOPOLOGY_NODE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定しなければなりません。

**num_entries**
戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**restore**
フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**nn_topology_node_detail.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。

**nn_topology_node_detail.node_name**
ネットワーク・トポロジー・データベースから得られたネットワーク修飾ノード名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）このフィールドは、APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**nn_topology_node_detail.node_type**
ノードのタイプ。以下のいずれかになります。

AP_NETWORK_NODE
AP_VRN

**nn_topology_node_detail.days_left**
トポロジー・データベースからこのノード項目を削除するまでの日数。ノードがローカル・ノードでない場合には、このフィールドはゼロより大きい値にセットされる , 要があります。

**nn_topology_node_detail.frsn**
フロー縮約シーケンス番号。これは q 源がローカル・ノードで更新された最終 ~ 刻を示します。

**nn_topology_node_detail.rsn**
q 源シーケンス番号。これは、この q 源を所有するネットワーク・ノードによって d り当てられます。

**nn_topology_node_detail.rar**
ネットワーク・ノードの経路追加レジスタンス。

**nn_topology_node_detail.status**
このフィールドはノードの状況を X 定し、AP_UNCONGESTED、または以下の内から 1 つまたは複数が互いに OR 結合されたものが可能です。

**AP_CONGESTED**
ISR セッションの数が START_NODE verb で X 定された **isr_sessions_upper_threshold** を越えています。

**AP_IRR_DEPLETED**
ISR セッションの数が START_NODE verb の **max_isr_sessions** パラメーターでX定された最大値に達しました。

**AP_ERR_DEPLETED**
エンドポイント・セッションの数がX定された最大数に到達しました。

**AP_QUIESCING**
タイプ AP_QUIENCE または AP_QUIENCE_ISR の STOP_NODE が発行されました。

**nn_topology_node_detail.function_support**
これはどの機能がサポートされるかをX定します。これは以下の内の 1 つまたは複数が可能です。

**AP_BORDER_NODE**
ボーダー・ノード・ファンクションはサポートされます。

**AP_CDS**
中央ディレクトリー・サーバーがサポートされます。

**AP_GATEWAY**
ノードはゲートウェイ・ノード (機能はアーキテクチャー的には未定義) です。

**AP_ISR**
このノードは中間セッション経路X定 (ISR) をサポートします。

**AP_HPR**
このノードは中間セッション経路X定 (ISR) をサポートします。

**AP_RTP_TOWER**
このノードは、HPR の RTP タワーをサポートします。

**AP_CONTROL_OVER_RTP_TOWER**
ノードは、RTP タワーを介した制御フローをサポートします。

**注:** AP_CONTROL_OVER_RTP_TOWER ノードは AP_HPR と AP_RTP_TOWER の両方の設定に対応します。

**node_type**
ノードのタイプ。以下のいずれかに設定されます。

AP_NETWORK_NODE
AP_VRN

**node_type** がT明の場合、AP_LEARN_NODE をX定する , 要があります。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**frsn** フロー縮約シーケンス番号。これが非ゼロの場合、この値以上の FRSN を } つトポロジー・リソースのみが戻されます。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**secondary_rc**
AP_INVALID_DAYS_LEFT

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DAYS_LEFT

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DAYS_LEFT

hte が関係する START_NODE パラメーターがセットされなかったために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

**secondary_rc**
AP_INVALID_DAYS_LEFT

ネットワーク・ノード・サポートをH用してシステムが構築されていなかったために verb が実行されない場合、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_INVALID_VERB

システム・エラーのために verb が実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# SFS_NN_TOPOLOGY_TG

**注:** この verb は SAFE_STORE_TOPOLOGY で取り代えられ、「プログラム」の直前バージョンとの互換性のためだけにリテインされます。

それぞれのネットワーク・ノードは、ネットワーク内にネットワーク・ノード、VRN、およびネットワーク・ノード間の TG に関する情報を保 } するネットワーク・トポロジー・データベースを保 } します。SFS_NN_TOPOLOGY_NODE は、ノードが再O動される場合に後でアクセス可能なトポロジー・データベース・ノード項目を安全に保管するためにH用されます。 **restore** フラグは情報が保管 (AP_NO) されるか、またはアクセス (AP_YES) されるかをX示するためにH用されます。verb は **topology_tg_detail** オーバーレーをH用します。

特定のネットワーク・ノードについての情報を得たり、いくつかの 『chunks』 のリスト情報を得るには、**owner**、**owner_type**、**dest**、**dest_type**、 と **tg_num** フィールドがセットされる , 要があります。

そうでない場合（ **list_options** フィールドに AP_FIRST_IN_LIST を設定する場合）、このフィールドは無 k されます。リスト形式のH用方法に関する背景知識については10ページの『ノードの照会』を 2 照してください。

このリストは **owner**、**owner_type**、**dest**、**dest_type**、 **tg_num** および **frsn** による配列です。**owner_type** と **dest** 名は最初に名前の長さ順による配列で、次に同じ長さの名前には ASCII 辞書編集の順序Uけになっています (IBM 6611 APPN MIB と一致)。 **owner_type** と **dest** は AP_ NETWORK_NODE、次に AP_VRN です。**tg_num** および **frsn** は数値順になります。

- AP_LIST_INCLUSIVE を選択すると、戻りリストはその名前の最初の有効なレコードから + O されます。
- AP_LIST_FROM_NEXT を選択すると、リストは、X定された名前の次の名前がついた最初の有効なレコードから + O されます。

**frsn** フィールドを非ゼロ値に設定すると、これより高いフロー縮約シーケンス番号 (FRSN) を } つデータベース項目のみが戻されることに注意してください。これは、まずノードの現行 FRSN を得ることによっていくつかの 『chunks』 内に一貫性のあるトポロジー・データベースを戻すことを可能にします。これは以下のように実行されます。

1. ノードの現行 FRSN を戻す QUERY_NODE を発行する。
2. , 要なだけ SFS_NN_TOPOLOGY_NODE（FRSN をゼロに設定して）を発行して、『chunks』 内のすべてのデータベース・エントリーをM得する。
3. QUERY_NODE を再発行して、新規作成 FRSN をステップ 1. で戻されたものと比 S する。
4. 2 つの FRSN が、データベース内で変更されたものと異なっている場合には、FRSN をステップ 1. で提供された FRSN より 1 だけ大きい値に設定して SFS_NN_TOPOLOGY_NODE を発行する。

## VCB 構造体

```
typedef struct sfs_nn_topology_tg
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   *buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;            /* buffer size                   */
        unsigned long   total_buf_size;      /* total buffer size required    */
        unsigned short  num_entries;         /* number of entries             */
        unsigned short  total_num_entries;   /* total number of entries       */
        unsigned char   list_options;        /* listing options               */
        unsigned char   restore;             /* store or restore;             */
        unsigned char   owner[17];           /* network qualified             */
                                             /* node name                     */
        unsigned char   owner_type;          /* node type                     */
        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                      */
        unsigned long   frsn;                /* flow-reduction sequence       */
                                             /* number                        */
} SFS_NN_TOPOLOGY_TG;

typedef struct nn_topology_tg_detail
{
        unsigned short  overlay_size;        /* size of this entry            */
        unsigned char   owner[17];           /* network qualified             */
        unsigned char   owner_type;          /* node type                     */
        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                      */
        unsigned long   frsn;                /* flow reduction sequence number */
        unsigned short  days_left            /* days left in database         */
        LINK_ADDRESS    dlc_data;            /* DLC signalling data           */
        unsigned long   rsn;                 /* resource sequence number      */
        unsigned char   status;              /* node status                   */
        TG_DEFINED_CHAR tg_chars;            /* TG characteristics            */
        unsigned char   reserva[20];         /* reserved                      */
}TOPOLOGY_TG_DETAIL;

typedef struct link_address
{
        unsigned short  length               /* length                        */
        unsigned short  reserve1;            /* reserved                      */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                             /* address                       */
}LINK_ADDRESS;
```

## 指定Ｑｉａー？ー

### restore = AP_NO のときの指定パラメーター

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_SFS_NN_TOPOLOGY_TG

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定し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。戻りデータはこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**list_options**
ここでは、リスト情報として何を戻すかを X 定します。X 定された **owner, owner_type, dest, dest_type, tg_num** と **frsn**（下のパラメーターを 2 照）は、戻される実際の情報の + O 点を示すために H 用される索引値を = します。

**AP_FIRST_IN_LIST**
索引値は無 k され、戻りリストはリスト内の最初の項目から O まります。

**AP_LIST_FROM_NEXT**
戻りリストは、システムに提供された索引値が示す項目の次の項目から + O されます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によって X 定される項目から O まります。

**restore**
フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**owner** TG の発信側ノードの名前（ , ずローカル・ノード名に設定されます)。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）このフィールドは、 APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**owner_type**
ノードのタイプ。以下のいずれかに設定されます。

AP_NETWORK_NODE
AP_VRN

**owner_type** が T 明の場合、AP_LEARN_NODE を X 定する , 要があります。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**dest** TG の完全修飾宛先ノード名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって

連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。) このフィールドは、APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**dest_type**
ノードのタイプ。以下のいずれかに設定されます。

AP_NETWORK_NODE
AP_VRN

**dest_type** が T 明の場合、 AP_LEARN_NODE を X 定する , 要があります。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**tg_num**
TG と関連した番号。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**frsn** フロー縮約シーケンス番号。これが非ゼロの場合、この値以上の FRSN を } つトポロジー・リソースのみが戻されます。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**buf_size**
バッファーに戻った情報の長さ。

**total_buf_size**
要求されたすべてのリスト情報を戻すために , 要になるバッファー・サイズを示す戻り値。これは **buf_size** の値より大きくできます。

**num_entries**
実際に戻された項目の数。

**total_num_entries**
戻すことができたはずの項目の合計数。これは **num_entries** の数より大きくできます。

**topology_tg_detail.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。

**topology_detail.owner**
TG の発信ノードの名前。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**topology_tg_detail.owner_type**
ノードのタイプ。以下のいずれかになります。

AP_NETWORK_NODE
AP_VRN

**topology_tg_detail.dest**
TG の完全修飾宛先ノード名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。) このフィールドは、APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**topology_tg_detail.dest_type**
ノードのタイプ。以下のいずれかになります。

AP_NETWORK_NODE
AP_VRN

**topology_tg_detail.tg_num**
TG と関連した番号。

**topology_tg_detail.frsn**
フロー縮約シーケンス番号。これは q 源がローカル・ノードで更新された最終 ~ 刻を示します。

**topology_tg_detail.days_left**
このノードの存在が N 認できない場合に、ノードが除去される前にトポロジー・データベースに D 留する日数。**owner** フィールドが X 定したノードがローカル・ノードでない場合には、このフィールドはゼロより大きい値にセットされる , 要があります。

**topology_tg_detail.dlc_data.length**
アドレスの長さ。

**topology_tg_detail.dlc_data.address**
アドレス。

**topology_tg_detail.rsn**
q 源シーケンス番号。これは、この q 源を所有するネットワーク・ノードによって d り当てられます。

**topology_tg_detail.status**
このフィールドは TG の状況を X 定します。これは以下の 1 つ、または複数が互いに OR 結合されたものが可能です。

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_TG_HPR
AP_TG_RTP
AP_NONE

**topology_tg_detail.tg_chars**
TG 特性。詳細については 33ページの『DEFINE_CN』を 2 照してください。

## 戻りＱｉａー？ー

パラメーター・エラーが原因で verb が正常に実行しない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_TG

AP_INVALID_ORIGIN_NODE
AP_INVALID_LIST_OPTION

verb が正常に実行されない場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

## 指定Ｑｉａー？ー

### restore = AP_YES のときの指定パラメーター

このアプリケーションは以下のパラメーターをX定します。

**opcode**
AP_SFS_NN_TOPOLOGY_TG

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報が書き込まれるバッファーへのポインター。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。その場合には、**buf_ptr** を NULL に設定しなければなりません。

**num_entries**
戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**buf_size**
バッファーに戻った情報の長さ。

**restore**
フラグは情報が | 元 (AP_YES) されるか、または保管 (AP_NO) されるかを示します。このケースでは、それは AP_NO にセットされます。

**total_num_entries**
戻すことができたはずの項目の合計数。これは **num_entries** の数より大きくできます。

**topology_tg_detail.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。これは、**restore** = AP_NO のときに戻される **overlay_size** と同じである , 要があります。

**topology_detail.owner**
TG の発信ノードの名前。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。( F 名前は組み込みスペースなしで、最大 8 バイトの長です。)

**topology_tg_detail.owner_type**
TG を所有するノードのタイプ。以下のいずれかになります。

AP_NETWORK_NODE
AP_VRN

**topology_tg_detail.dest**
TG の完全修飾宛先ノード名。この名前は 17 バイトの隣接制御点の名前です。右側には EBCDIC スペースが埋め込まれます。EBCDIC ドットによって連結された 2 つのタイプ A EBCDIC 文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）このフィールドは、APPN ノードへのリンクの場合にのみ適切であり、それ以 0 の場合には無 k されます。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

**topology_tg_detail.dest_type**
ノードのタイプ。以下のいずれかになります。

AP_NETWORK_NODE
AP_VRN

**topology_tg_detail.tg_num**
TG と関連した番号。

**topology_tg_detail.frsn**
フロー縮約シーケンス番号。これは q 源がローカル・ノードで更新された最終 ~ 刻を示します。

**topology_tg_detail.days_left**
このノードの存在が N 認できない場合に、ノードが除去される前にトポロジー・データベースに D 留する日数。**owner** フィールドが X 定したノードがローカル・ノードでない場合には、このフィールドはゼロより大きい値にセットされる , 要があります。

**topology_tg_detail.dlc_data.length**
アドレスの長さ。

**topology_tg_detail.dlc_data.address**
アドレス。

**topology_tg_detail.rsn**
q 源シーケンス番号。これは、この q 源を所有するネットワーク・ノードによって d り当てられます。

**topology_tg_detail.status**
このフィールドは TG の状況をX定します。これは以下に示す 1 つまたは複数が互いに OR 結合されたものが可能です。

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_TG_HPR
AP_TG_RTP
AP_NONE

**topology_tg_detail.tg_chars**
TG 特性。詳細については 33ページの『DEFINE_CN』を 2 照してください。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_DAYS_LEFT

関係のある START_NODE パラメーターがセットされなかったために verb が実行されない場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ネットワーク・ノード・サポートをH用してシステムが構築されなかったために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_INVALID_VERB

ノードが + O されていないために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSYEM_ERROR

**SFS_NN_TOPOLOGY_TG**

# 第8章 ; C 7 g s限度 verb

この章では、セッション限度の初期設定、変更、リセットにH用する verb について説明します。

# CHANGE_SESSION_LIMIT

CHANGE_SESSION_LIMIT verb では、それぞれのモード（またはセッション・グループ）のセッション限度の変更を要求します。この verb の処理結果として、セッションが h 動化されたり非 h 動化されたりすることがあります。

## VCB 構造体

```
typedef struct change_session_limit
{
   unsigned short  opcode;         /* verb operation code       */
   unsigned char   reserv2;        /* reserved                  */
   unsigned char   format;         /* format                    */
   unsigned short  primary_rc;     /* primary return code       */
   unsigned long   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? */
   unsigned short  plu_mode_session_limit;
                                   /* session limit             */
   unsigned short  min_conwinners_source;
                                   /* min source contention     */
                                   /* winner sessions           */
   unsigned short  min_conwinners_target;
                                   /* min target contention     */
                                   /* winner sessions           */
   unsigned short  auto_act;       /* auto activation limit     */
   unsigned char   responsible;    /* responsible indicator     */
   unsigned char   reserv4[3];     /* reserved                  */
   unsigned long   sense_data;     /* sense data                */
} CHANGE_SESSION_LIMIT;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_CHANGE_SESSION_LIMIT

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

セッション限度の変更を要求するローカル LU の LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文ｚ列です。このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、**lu_alias** フィールドがH用されます。

**lu_alias**

セッション限度の変更を要求するローカル LU の別名。これは、ローカル＝示可能文ｚセットの 8 バイトの文ｚ列です。このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトす

べてが意味を } つので、8 バイトすべてを設定する , 要があります。**lu_name** フィールドと **lu_alias** フィールドを両方ともすべてゼロに設定すると、verb は、制御点に関連Uけられている LU（省略 ~ の LU）に転送されます。

**plu_alias**

ローカル LU がパートナー LU を識別するための名前。この名前は、構成 ~ にX定したパートナー LU の名前と一致していなければなりません。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する , 要があります。このフィールドをすべてゼロに設定すると、**fqplu_name** フィーリ

セッション数がこの限度以下で通常の操作（AP_DEACT_NORMAL をX定して）により非h動化されたときには、新規作成セッションがこの限度までh動化されます。

**responsible**
セッション限度の変更後に、ソース（ローカル）LU とターゲット（パートナー）LU のどちらがセッションの非h動化を行うかをX定します（AP_SOURCE または AP_TARGET）。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**secondary_rc**
AP_AS_SPECIFIED

AP_AS_NEGOTIATED

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_LU_MODE_SESSION_LIMIT_ZERO

AP_EXCEEDS_MAX_ALLOWED
AP_INVALID_MODE_NAME
AP_INVALID_PLU_NAME
AP_INVALID_RESPONSIBLE
AP_INVALID_SET_NEGOTIABLE
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_MODE_RESET

ノードがまだ+Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停_しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

d り振りエラーのために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_ALLOCATION_ERROR

**secondary_rc**
AP_ALLOCATION_FAILURE_NO_RETRY

**sense_data**
d り振りエラーに関連したセンス・データ。

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

エラーのために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_CONV_FAILURE_NO_RETRY

AP_CNOS_PARTNER_LU_REJECT

**secondary_rc**
AP_CNOS_COMMAND_RACE_REJECT

AP_CNOS_MODE_NAME_REJECT

# INITIALIZE_SESSION_LIMIT

INITIALIZE_SESSION_LIMIT verb では、モードのセッション限度を初期化します。

## VCB 構造体

```
typedef struct initialize_session_limit
{
   unsigned short  opcode;         /* verb operation code       */
   unsigned char   reserv2;        /* reserved                  */
   unsigned char   format;         /* format                    */
   unsigned short  primary_rc;     /* primary return code       */
   unsigned long   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? */
   unsigned short  plu_mode_session_limit;
                                   /* session limit             */
   unsigned short  min_conwinners_source;
                                   /* min source contention     */
                                   /* winner sessions           */
   unsigned short  min_conwinners_target;
                                   /* min target contention     */
                                   /* winner sessions           */
   unsigned short  auto_act;       /* auto activation limit     */
   unsigned char    reserv4[4];    /* reserved                   */
   unsigned long   sense_data;     /* sense data                 */
} INITIALIZE_SESSION_LIMIT;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_INITIALIZE_SESSION_LIMIT

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

セッション限度の初期化を要求するローカル LU の LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文z列です。このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、**lu_alias** フィールドがH用されます。

**lu_alias**

セッション限度の初期化を要求するローカル LU の別名。これは、ローカル=示可能文z セットの 8 バイトの文z列です。このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を } つので、8 バイトすべてを設定する , 要があります。

**lu_name** フィールドと **lu_alias** フィールドを両方ともすべてゼロに設定すると、verb は、制御点に関連Uけられている LU（省略~の LU）に転送されます。

**plu_alias**
ローカル LU がパートナー LU を識別するための名前。この名前は、構成~にX定したパートナー LU の名前と一致していなければなりません。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する,要があります。このフィールドをすべてゼロに設定すると、**fqplu_name** フィールドが,須パートナー LU をX定するためにH用されます。

**fqplu_name**
パートナー LU の完全修飾 LU 名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）このフィールドは、 **plu_alias** フィールドにすべてゼロを設定した場合にのみ有効です。

**mode_name**
構成~に定義されたネットワーキング特性セットの名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

この verb は、モード名 SNASVCMG または CPSVCMG がこのフィールドでシステムに提供され、限&値が **plu_mode_session_limit 2** 、**min_conwinners_source 1**、 と **min_conwinners target 1** 以 0 の値を取る場合には、リジェクトされます。

**set_negotiable**
このモードの最大交渉可能セッション限度を **plu_mode_session_limit** の値に修正するかどうかをX定します。

AP_YES
AP_NO

**plu_mode_session_limit**
このモードで要求した合計セッション限度。実際のセッション限度（パートナー LU との間で交渉可能）は、このモードのローカル LU とパートナー LU の間で合意したサポート範囲内の最大セッション数になります。ここには、1 から 32 767 の範囲の値を設定する,要があります。

**min_conwinners_source**
このモードでローカル LU が競合勝者になる最小セッション数。ここには、0 から 32 767 の範囲の値を設定する,要があります。

**min_conwinners_target**
このモードでパートナー LU が競合勝者になる最小セッション数。ここには、0 から 32 767 の範囲の値を設定する,要があります。

**auto_act**
セッション限度の変更後に自動的に h 動化されるセッションの数。自動的に h 動化されるセッションの実際の数は、ここでX定した値と、交渉後のロー

カル LU の競合勝者セッション最小数のうち、小さいほうの値になります。セッション数がこの限度以下で通常の操作（ AP_DEACT_NORMAL をX定して）により非 h 動化されたときには、新規作成セッションがこの限度まで h 動化されます。ここには、0 から 32 767 の範囲の値を設定する , 要があります。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**secondary_rc**
AP_AS_SPECIFIED

AP_AS_NEGOTIATED

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_CANT_CHANGE_TO_ZERO

AP_EXCEEDS_MAX_ALLOWED
AP_INVALID_SET_NEGOTIABLE
AP_INVALID_PLU_NAME
AP_INVALID_MODE_NAME
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_SCVMG_LIMITS

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_MODE_NOT_RESET

ノードがまだ + O されていないために verb が実行されない場合には、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

d り振りエラーのために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_ALLOCATION_ERROR

**secondary_rc**
AP_ALLOCATION_FAILURE_NO_RETRY

**sense_data**
d り振りエラーに関連したセンス・データ。

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

エラーのために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_CONV_FAILURE_NO_RETRY

AP_CNOS_PARTNER_LU_REJECT

**secondary_rc**
AP_CNOS_COMMAND_RACE_REJECT

AP_CNOS_MODE_NAME_REJECT

# RESET_SESSION_LIMIT

RESET_SESSION_LIMIT verb では、モードのセッション限度のリセットを要求します。

## VCB 構造体

```
typedef struct reset_session_limit
{
   unsigned short  opcode;           /* verb operation code        */
   unsigned char   reserv2;          /* reserved                   */
   unsigned char   format;           /* format                     */
   unsigned short  primary_rc;       /* primary return code        */
   unsigned long   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 qual 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?  */
   unsigned char   reserv4[8];       /* reserved                   */
   unsigned char   responsible;      /* responsible                */
   unsigned char   drain_source;     /* drain source               */
   unsigned char   drain_target;     /* drain target               */
   unsigned char   force;            /* force                      */
   unsigned long   sense_data;       /* sense data                 */
} RESET_SESSION_LIMIT;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_RESET_SESSION_LIMIT

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

セッション限度のリセットを要求するローカル LU の LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文z列です。このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、 **lu_alias** フィールドがH用されます。

**lu_alias**

セッション限度のリセットを要求するローカル LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を}つので、8 バイトすべてを設定する,要があります。ここにすべてゼロを設定すると、verb は、制御点に関連Uけられている LU（省略~の LU）に転送されます。

**plu_alias**

ローカル LU がパートナー LU を識別するための名前。この名前は、構成~にX定したパートナー LU の名前と一致していなければなりません。これは、

ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する , 要があります。このフィールドをすべてゼロに設定すると、**fqplu_name** フィールドが , 須パートナー LU を X 定するために H 用されます。

**fqplu_name**
パートナー LU の完全修飾 LU 名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。) このフィールドは、**plu_alias** フィールドにすべてゼロを設定した場合にのみ有効です。

**mode_name**
構成 ~ に定義されたネットワーキング特性セットの名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列 (先頭は文 z ) です。EBCDIC スペースが右の余白に埋め込まれます。

**mode_name_select**
1 つの X 定モードのセッション限度をリセットするか、ローカル LU とパートナー LU の間のすべてのモードのセッション限度をリセットするかを選択します。

AP_ONE
AP_ALL

**set_negotiable**
このモードの最大交渉可能セッション限度を修正するかどうかを X 定します。

AP_YES
AP_NO

**responsible**
セッション限度のリセット後に、ソース (ローカル) LU とターゲット (パートナー) LU のどちらがセッションの非 h 動化を行うかを X 定します (AP_SOURCE または AP_TARGET)。

**drain_source**
セッション限度を変更またはリセットするとき、ソース LU がセッションを非 h 動化する前に、待機中のセッション要求を処理するかどうかを X 定します (AP_NO または AP_YES)。

**drain_target**
セッション限度を変更またはリセットするとき、ターゲット LU がセッションを非 h 動化する前に、待機中のセッション要求を処理するかどうかを X 定します (AP_NO または AP_YES)。

**force** CNOS 交渉が正常に実行されなかった場合でも、セッション限度をゼロに設定するかどうかを X 定します (AP_YES または AP_NO)。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

## RESET_SESSION_LIMIT

**primary_rc**
AP_OK

**secondary_rc**
AP_FORCED

AP_AS_SPECIFIED
AP_AS_NEGOTIATED

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_EXCEEDS_MAX_ALLOWED

AP_INVALID_PLU_NAME
AP_INVALID_MODE_NAME
AP_INVALID_MODE_NAME_SELECT
AP_INVALID_RESPONSIBLE
AP_INVALID_DRAIN_SOURCE
AP_INVALID_DRAIN_TARGET
AP_INVALID_FORCE
AP_INVALID_SET_NEGOTIABLE
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_MODE_RESET

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

d り振りエラーのために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_ALLOCATION_ERROR

**secondary_rc**
AP_ALLOCATION_FAILURE_NO_RETRY

**sense_data**
d り振りエラーに関連したセンス・データ。

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

エラーのために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_CONV_FAILURE_NO_RETRY

AP_CNOS_PARTNER_LU_REJECT

**secondary_rc**
AP_CNOS_COMMAND_RACE_REJECT

AP_CNOS_MODE_NAME_REJECT

# 第9章 Ｎ－Ｉ・＊Ｚｌ－？－機能 API の指示

ノード・オペレーター機能 API は、ノード内の変化をノード・オペレーターに通知するためのX示 verb を生成します。 verb の一般的な構造は、以下のようになります。

```
typedef struct indication_hdr
          {
            unsigned short  opcode;         /* verb operation code           */
            unsigned char   reserv2;        /* reserved                      */
            unsigned char   format;         /* format                        */
            unsigned short  primary_rc;     /* primary return code           */
            unsigned long   secondary_rc;   /* secondary return code         */
            unsigned char   data_lost;      /* previous indication lost      */
          } INDICATION_HDR;
```

# DLC_INDICATION

このX示は、DLC がアクティブから非アクティブになった~、または非アクティブからアクティブになった~に生成されます。

## VCB 構造体

```
typedef struct dlc_indication
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   attributes;          /* verb attributes                */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   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[RD_LEN]; /* resource description           */
        unsigned char   reserva[20];         /* reserved                       */
} DLC_INDICATION;
```

## Ｑｉａ一？一

**opcode**

AP_DLC_INDICATION

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。**data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する，要があります。

**deactivated**

DLC が非アクティブになった場合は AP_YES に設定され、 DLC がアクティブになった場合は AP_NO に設定されます。

**dlc_name**

DLC 名。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**description**

リソース記述（DEFINE_DLC でX定）。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

# DLUR_LU_INDICATION

このX示は、 DLUR LU がh動化または非h動化された~に生成されます。このX示によって、登録済みのアプリケーションは、現行のアクティブ DLUR LU のリストをメンテナンスすることができます。

## VCB 構造体

```
typedef struct dlur_lu_indication
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   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;
```

## Ｑｉａ－？－

**opcode**

AP_DLUR_LU_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障2を検出した場合に設定されます。**data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する，要があります。

**reason**

DLUR LU が DLUS によってh動化された場合は AP_ADDED に設定されます。 DLUR LU が DLUS によって明示的に非h動化された場合、またはリンク障2や PU の非h動化によって暗黙に非h動化された場合は、AP_REMOVED に設定されます。

**lu_name**

LU 名。これは 8 バイトの英数zタイプ A の EBCDIC 文z列（先頭は英z）で、右側の余白には EBCDIC スペースを埋め込みます。

**pu_name**

この LU がH用する PU の名前。これは 8 バイトの英数 z タイプ A の EBCDIC 文 z 列（先頭は英 z ）で、右側の余白には EBCDIC スペースを埋め込みます。

**nau_address**

LU のネットワーク・アドレス可能単位のアドレス。1 〜 255 の範囲になります。

# DLUR_PU_INDICATION

このX示は、DLUR PU がh動化または非h動化された~に生成されます。 このX示によって、登録済みのアプリケーションは現行のアクティブ DLUR LU のリストをメンテナンスすることができます。

## VCB 構造体

```
typedef struct dlur_pu_indication
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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;
```

## Qia ー？ー

**opcode**

AP_DLUR_PU_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。**data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために 1 つの照会 verb を発行する , 要があります。

**reason**

X示の原因。以下のいずれかになります。

**AP_ACTIVATION_STARTED**

PU はh動化しています。

**AP_ACTIVATING**

PU がアクティブになりました。

**AP_DEACTIVATING**

PU が非アクティブになりました。

**AP_FAILED**

PU は失敗しました。

**AP_ACTIVATION_FAILED**
PU はh動化に失敗しました。

**pu_name**
PU の名前 。これは 8 バイトの英数zタイプ A の EBCDIC 文z列（先頭は英z）で、右側の余白には EBCDIC スペースを埋め込みます。

**pu_id** DEFINE_INTERNAL_PU verb で定義されている PU 識別R、またはダウンストリーム PU から XID で取得される PU 識別R。これは、4 バイトの 16 進数文z列です。ビット 0 〜 11 にはブロック番号が設定され、ビット 12 〜 31 には PU を固有に識別する ID 番号が設定されます。

**plu_location**
PU の名前 。これは以下のうちのどれかになります。

AP_INTERNAL
AP_DOWNSTREAM

**dlur_pu_detail.pu_status**
PU の状況（DLUR から見た)。これには以下の値の 1 つを設定することが可能です。

**AP_RESET_NO_RETRY**
PU はリセット状態にあり、再n行されません。

**AP_RESET_RETRY**
PU はリセット状態にあり、再n行されます。

**AP_PEND_ACTPU**
PU はホストからの ACTPU を待っています。

**AP_PEND_ACTPU_RSP**
DLUR は、ACTPU を PU に転送した後、PU からの応答を待っています。

**AP_ACTIVE**
PU はアクティブです。

**AP_PEND_DACTPU_RSP**
DLUR は、DACTPU を PU に転送した後、PU からの応答を待っています。

**AP_PEND_INOP**
DLUR は PU を非h動化にする前に、完了しなければならないすべてのイベントの完了を待っています。

**pu_id** PU が現在H用している (または、H用をn行している) DLUS ノード名。これは 17 バイトの文z列で、2 つのタイプ A EBCDIC 文z列を EBCDIC ドットで区切った形になります。右側の余白は、EBCDIC のスペースを埋めます。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）PU h動化が失敗した場合、このフィールドはすべてゼロに設定されます。

**dlur_pu_detail.dlus_session_status**
現在、PU によってH用されている DLUS パイプの状況。これは以下のうちのどれかになります。

AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE
AP_INACTIVE

# DLUS_INDICATION

このX示は、 DLUS ノードへのパイプが非アクティブからアクティブに（またはその逆に）なった ~ に生成されます。パイプが非アクティブになった場合は、パイプの統計が = 示されます。

## VCB 構造体

```
typedef struct dlus_indication
{
        unsigned short  opcode;             /* verb operation code            */
        unsigned char   reserv2;            /* reserved                       */
        unsigned char   format;             /* format                         */
        unsigned short  primary_rc;         /* primary return code            */
        unsigned long   secondary_rc;       /* secondary return code          */
        unsigned char   data_lost;          /* previous indication lost       */
        unsigned char   deactivated;        /* has session been deactivated?  */
        unsigned char   dlus_name[17];      /* DLUS name                      */
        unsigned char   reserv1;            /* reserved                       */
        PIPE_STATS      pipe_stats;         /* pipe statistics                */
        unsigned char   reserva[20];        /* reserved                       */
} DLUS_INDICATION;

typedef struct pipe_stats
{
        unsigned long   reqactpu_sent;      /* REQACTPUs sent to DLUS         */
        unsigned long   reqactpu_rsp_received;
                                            /* RSP(REQACTPU)s received        */
                                            /* from DLUS                      */
        unsigned long   actpu_received;     /* ACTPUs received from DLUS      */
        unsigned long   actpu_rsp_sent;     /* RSP(ACTPU)s sent to DLUS       */
        unsigned long   reqdactpu_sent;     /* REQDACTPUs sent to DLUS        */
        unsigned long   reqdactpu_rsp_received;
                                            /* RSP(REQDACTPU)s received       */
                                            /* from DLUS                      */
        unsigned long   dactpu_received;    /* DACTPUs received from DLUS     */
        unsigned long   dactpu_rsp_sent;    /* RSP(DACTPU)s sent to DLUS      */
        unsigned long   actlu_received;     /* ACTLUs received from DLUS      */
        unsigned long   actlu_rsp_sent;     /* RSP(ACTLU)s sent to DLUS       */
        unsigned long   dactlu_received;    /* DACTLUs received from DLUS     */
        unsigned long   dactlu_rsp_sent;    /* RSP(DACTLU)s sent to DLUS      */
        unsigned long   sscp_pu_mus_rcvd;   /* MUs for SSCP-PU sess received  */
        unsigned long   sscp_pu_mus_sent;   /* MUs for SSCP-PU sessions sent  */
        unsigned long   sscp_lu_mus_rcvd;   /* MUs for SSCP-LU sess received  */
        unsigned long   sscp_lu_mus_sent;   /* MUs for SSCP-LU sessions sent  */
} PIPE_STATS;
```

## Ｑｉａ－？－

**opcode**

AP_DLUS_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

## DLUS_INDICATION

**data_lost**
データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前の X 示が失われた障 2 を検出した場合に設定されます。**data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**deactivated**
パイプが非アクティブになった場合は AP_YES に設定され、パイプがアクティブになった場合は AP_NO に設定されます。

**dlus_name**
DLUS 名。これは 17 バイトの文 z 列で、2 つのタイプ A EBCDIC 文 z 列を EBCDIC ドットで区切った形になります。右側の余白は、EBCDIC スペースを埋めます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**pipe_stats.reqactpu_sent**
パイプを介して DLUS に送られた REQACTPU の数。

**pipe_stats.reqactpu_rsp_received**
パイプを介して DLUS から受信された RSP (REQACTPU) の数。

**pipe_stats.actpu_received**
パイプを介して DLUS から受信された ACTPU の数。

**pipe_stats.actpu_rsp_sent**
パイプを介して DLUS に送られた RSP (ACTPU) の数。

**pipe_stats.reqdactpu_sent**
パイプを介して DLUS に送られた REQDACTPU の数。

**pipe_stats.reqdactpu_rsp_received**
パイプを介して DLUS から受信された RSP (REQDACTPU) の数。

**pipe_stats.dactpu_received**
パイプを介して DLUS から受信された DACTPU の数。

**pipe_stats.dactpu_rsp_sent**
パイプを介して DLUS に送られた RSP (DACTPU) の数。

**pipe_stats.actlu_received**
パイプを介して DLUS から受信された ACTLU の数。

**pipe_stats.actlu_rsp_sent**
パイプを介して DLUS に送られた RSP (ACTLU) の数。

**pipe_stats.dactlu_received**
パイプを介して DLUS から受信するされた DACTLU の数。

**pipe_stats.dactlu_rsp_sent**
パイプを介して DLUS に送られた RSP (DACTLU) の数。

**pipe_stats.sscp_pu_mus_rcvd**
パイプを介して DLUS から受信するされた SSCP-PU MU の数。

**pipe_stats.sscp_pu_mus_sent**
パイプを介して DLUS に送られた SSCP-PU MU の数。

**pipe_stats.sscp_lu_mus_rcvd**
パイプを介して DLUS から受信された SSCP-LU MU の数。

**pipe_stats.sscp_lu_mus_sent**
パイプを介して DLUS に送られた SSCP-LU MU の数。

# DOWNSTREAM_LU_INDICATION

この verb はCommunications Serverにのみ適用します。

このX示は、ダウンストリーム LU とホストの間の LU-SSCP セッションが非アクティブからアクティブに（またはその逆に）なった～、または PLU-SLU セッションが非アクティブからアクティブに（またはその逆に）なった～に生成されます。LU-SSCP セッションが非 h 動化された～は、LU-SSCP 統計が提供され、PLU-SLU セッションが非 h 動化された～は、PLU-SLU 統計が提供されます。

## VCB 構造体

```
typedef struct downstream_lu_indication
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   attributes;          /* attributes                     */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   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[RD_LEN]; /* resource description           */
        unsigned char   nau_address;         /* NAU address                    */
        unsigned char   lu_sscp_sess_active; /* Is 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 sess stats  */
        SESSION_STATS   us_plu_stats;        /* Upstream PLU-SLU sess stats    */
} DOWNSTREAM_LU_INDICATION;

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size        */
        unsigned short  send_ru_size;        /* session send RU size           */
        unsigned short  max_send_btu_size;   /* max send BTU size              */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size               */
        unsigned short  max_send_pac_win;    /* max send pacing window size    */
        unsigned short  cur_send_pac_win;    /* curr send pacing window size   */
        unsigned short  max_rcv_pac_win;     /* max rcv pacing window size     */
        unsigned short  cur_rcv_pac_win;     /* curr receive pacing winsize    */
        unsigned long   send_data_frames;    /* number of data frames sent     */
        unsigned long   send_fmd_data_frames;
                                             /* num FMD data frames sent       */
        unsigned long   send_data_bytes;     /* number of data bytes sent      */
        unsigned long   rcv_data_frames;     /* num of data frames received    */
        unsigned long   rcv_fmd_data_frames; /* num FMD data frames received   */
        unsigned long   rcv_data_bytes;      /* num 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[8];          /* Link station name              */
        unsigned char   pacing_type;         /* type of pacing in use          */
} SESSION_STATS;
```

## Q i a ー ? ー

**opcode**

AP_DOWNSTREAM_LU_INDICATION

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前の X 示が失われた障 2 を検出した場合に設定されます。**data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**dspu_name**

ダウンストリーム LU に関連 U 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、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**ls_name**

リンク・ステーションの名前。 これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する , 要があります。

**dslu_name**

ダウンストリーム LU 名。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**description**

リソースの記述（DEFINE_DOWNSTREAM_LU で X 定）。

**nau_address**

LU のネットワーク・アドレス可能単位のアドレス。1 ～ 255 の範囲になります。

**lu_sscp_sess_active**

ダウンストリーム LU との LU-SSCP セッションがアクティブであるかどうかを示します。 AP_YES または AP_NO のいずれかを設定します。

## DOWNSTREAM_LU__INDICATION

**plu_sess_active**
ダウンストリーム LU との PLU-SLU セッションがアクティブであるかどうかを示します。 AP_YES または AP_NO のいずれかを設定します。

**dspu_services**
ローカル・ノードがリンクを介してダウンストリーム LU に提供するサービスを示します。 以下のいずれかの値に設定されます。

**AP_PU_CONCENTRATION**
ローカル・ノードは、ダウンストリーム PU に対して PU 集信を提供します。

**AP_DLUR**
ローカル・ノードは、ダウンストリーム PU に対して DLUR サポートを提供します。

**lu_sscp_stats.rcv_ru_size**
このフィールドは常に予約済みです。

**lu_sscp_stats.send_ru_size**
このフィールドは常に予約済みです。

**lu_sscp_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**lu_sscp_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**lu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.cur_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.send_data_frames**
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_frames**
送信された通常フロー FMD データ・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・バイト数。

**lu_sscp_stats.rcv_data_frames**
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_frames**
受信された通常フロー FMD データ・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・バイト数。

**lu_sscp_stats.sidh**
セッション ID 上位バイト。

**lu_sscp_stats.sidl**
セッション ID の下位バイト。

**lu_sscp_stats.odai**
原点宛先アドレスX示。セッションを立ち上げた~、ローカル・ノードに 1 次リンク・ステーションが入っている場合にはバインドの送信側はフィールドをゼロにセットします。また、バインドの送信側が 2 次リンク・ステーションが入ったノードの場合には、フィールドを 1 にセットします。

**lu_sscp_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**lu_sscp_stats.pacing_type**
アップストリーム LU-SSCP セッションでH用中の受信ペーシングのタイプ。これは値 AP_NONE をとります。

**ds_plu_stats.rcv_ru_size**
最大の受信 RU サイズ。

**ds_plu_stats.send_ru_size**
最大の送信 RU サイズ。

**ds_plu_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**ds_plu_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**ds_plu_stats.max_send_pac_win**
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。

**ds_plu_stats.cur_send_pac_win**
このセッションの送信ペーシング・ウィンドウの現行サイズ。

**ds_plu_stats.max_rcv_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。

**ds_plu_stats.cur_rcv_pac_win**
このセッションの受信ペーシング・ウィンドウの現行サイズ。

**ds_plu_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**ds_plu_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**ds_plu_stats.send_data_bytes**
送信された通常フロー・データ・バイト数。

**ds_plu_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**ds_plu_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**ds_plu_stats.rcv_data_bytes**
受信された通常フロー・データ・バイト数。

## DOWNSTREAM_LU__INDICATION

**ds_plu_stats.sidh**
セッション ID 上位バイト。

**ds_plu_stats.sidl**
セッション ID の下位バイト。

**ds_plu_stats.odai**
原点宛先アドレスX示。セッションを立ち上げた~、ローカル・ノードに 1 次リンク・ステーションが入っている場合にはバインドの送信側はフィールドをゼロにセットします。また、バインドの送信側が 2 次リンク・ステーションが入ったノードの場合には、フィールドを 1 にセットします。

**ds_plu_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**ds_plu_sscp_stats.pacing_type**
ダウンストリーム PLU-SLU セッションでH用中の受信ペーシングのタイプ。これは AP_NONE または AP_PACING_FIXED をセットできます。

**us_plu_stats.rcv_ru_size**
最大の受信 RU サイズ。

**us_plu_stats.send_ru_size**
最大の送信 RU サイズ。

**us_plu_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**us_plu_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**us_plu_stats.max_send_pac_win**
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。

**us_plu_stats.cur_send_pac_win**
このセッションの送信ペーシング・ウィンドウの現行サイズ。

**us_plu_stats.max_rcv_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。

**us_plu_stats.cur_rcv_pac_win**
このセッションの受信ペーシング・ウィンドウの現行サイズ。

**us_plu_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**us_plu_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**us_plu_stats.send_data_bytes**
送信された通常フロー・データ・バイト数。

**us_plu_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**us_plu_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**us_plu_stats.rcv_data_bytes**
受信された通常フロー・データ・バイト数。

**us_plu_stats.sidh**
セッション ID 上位バイト。**dspu_services** に AP_PU_CONCENTRATION が設定されている場合、このフィールドは予約済みになります。

**us_plu_stats.sidl**
セッション ID の下位バイト。**dspu_services** に AP_PU_CONCENTRATION が設定されている場合、このフィールドは予約済みになります。

**us_plu_stats.odai**
原点宛先アドレスX示。セッションを立ち上げた~、ローカル・ノードに 1 次リンク・ステーションが入っている場合にはバインドの送信側はフィールドをゼロにセットします。また、バインドの送信側が 2 次リンク・ステーションが入ったノードの場合には、フィールドを 1 にセットします。**dspu_services** に AP_PU_CONCENTRATION が設定されている場合、このフィールドは予約済みになります。

**us_plu_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。8 バイトすべてに意味があります。**dspu_services** に AP_PU_CONCENTRATION が設定されている場合、このフィールドは予約済みになります。

**us_plu_stats.pacing_type**
アップストリーム PLU-SLU セッションでH用中の受信ペーシングのタイプ。これは値 AP_NONE または AP_PACING_FIXED を取れます。

# DOWNSTREAM_PU_INDICATION

この verb はCommunications Serverにのみ適用します。

このX示は、ダウンストリーム PU とホストの間の PU-SSCP セッションが非アクティブからアクティブに（またはその逆に）なった~に生成されます。PU-SSCP セッションが非 h 動化された~は、PU-SSCP 統計が = 示されます。

## VCB 構造体

```
typedef struct downstream_pu_indication
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* attributes                    */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   data_lost;           /* previous indication lost      */
        unsigned char   dspu_name[8];        /* PU Name                       */
        unsigned char   description[RD_LEN]; /* resource description          */
        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
{
        unsigned short  rcv_ru_size;         /* session receive RU size       */
        unsigned short  send_ru_size;        /* session send RU size          */
        unsigned short  max_send_btu_size;   /* max send BTU size             */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size              */
        unsigned short  max_send_pac_win;    /* max send pacing window size   */
        unsigned short  cur_send_pac_win;    /* curr send pacing window size  */
        unsigned short  max_rcv_pac_win;     /* max rcv pacing window size    */
        unsigned short  cur_rcv_pac_win;     /* curr receive pacing winsize   */
        unsigned long   send_data_frames;    /* number of data frames sent    */
        unsigned long   send_fmd_data_frames;
                                             /* num FMD data frames sent      */
        unsigned long   send_data_bytes;     /* number of data bytes sent     */
        unsigned long   rcv_data_frames;     /* num of data frames received   */
        unsigned long   rcv_fmd_data_frames; /* num FMD data frames received  */
        unsigned long   rcv_data_bytes;      /* num 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[8];          /* Link station name             */
        unsigned char   pacing;              /* pacing_type                   */
} SESSION_STATS;
```

## Ｑｉａ一？一

**opcode**

AP_DOWNSTREAM_PU_INDICATION

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**
AP_OK

**secondary_rc**
ゼロになります。

**data_lost**
データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。**data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**dspu_name**
ダウンストリーム PU の名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**description**
リソースの記述（DEFINE_LS でX定）。

**ls_name**
リンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**pu_sscp_sess_active**
ダウンストリーム PU との PU-SSCP セッションがアクティブであるかどうかを示します。 AP_YES または AP_NO のいずれかを設定します。

**dspu_services**
ローカル・ノードがリンクを介してダウンストリーム PU に提供するサービスを示します。 以下のいずれかの値に設定されます。

**AP_PU_CONCENTRATION**
ローカル・ノードは、ダウンストリーム PU に対して PU 集信を提供します。

**AP_DLUR**
ローカル・ノードは、ダウンストリーム PU に対して DLUR サポートを提供します。

**pu_sscp_stats.rcv_ru_size**
このフィールドは常に予約済みです。

**pu_sscp_stats.send_ru_size**
このフィールドは常に予約済みです。

**pu_sscp_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

## DOWNSTREAM_PU__INDICATION

**pu_sscp_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**pu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_sscp_stats.cur_rcv_pac_win**
このフィールドは、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_frames**
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_frames**
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・バイト数。

**pu_sscp_stats.sidh**
セッション ID 上位バイト。

**pu_sscp_stats.sidl**
セッション ID の下位バイト。

**pu_sscp_stats.odai**
原点宛先アドレスX示。セッションを立ち上げた~、ローカル・ノードに 1 次リンク・ステーションが入っている場合にはバインドの送信側はフィールドをゼロにセットします。また、バインドの送信側が 2 次リンク・ステーションが入ったノードの場合には、フィールドを 1 にセットします。

**pu_sscp_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**pu_sscp_stats.pacing_type**
アップストリーム PU-SSCP セッションでH用中の受信ペーシングのタイプ。これは値 AP_NONE をとります。

# FOCAL_POINT_INDICATION

このX示は、フォーカル・ポイントのM得、変更、または取り消しが行われた~に生成されます。

## VCB 構造体

```
typedef struct focal_point_indication
{
        unsigned short  opcode;              /* verb operation code            */
        unsigned char   reserv2;             /* reserved                       */
        unsigned char   format;              /* format                         */
        unsigned short  primary_rc;          /* primary return code            */
        unsigned long   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;
```

## Ｑｉａ－？－

**opcode**

AP_FOCAL_POINT_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO)。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。**data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**ms_category**

M得、変更、または取り消しが行われたフォーカル・ポイントのカテゴリー。この値は、「*SNA Management Services*」で説明された管理サービス・カテゴリーの 4 バイト構造定義値（右側の余白は EBCDIC のスペースで埋められる) の中の 1 つか、または 8 バイトのタイプ 1134 の EBCDIC 導入定義名のいずれかになります。

**fp_fqcp_name**

現行フォーカル・ポイントの完全修飾 CP 名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによっ

て連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）この名前は、フォーカル・ポイントが取り消され、まだ置換されていない場合（したがって、現 ~ 点でアクティブなフォーカル・ポイントがない場合）に、すべてゼロになります。

**ms_appl_name**
現行フォーカル・ポイントのアプリケーション名。この値は、管理サービス・アプリケーションの 4 バイト構造定義値（右側の余白は EBCDIC のスペースで埋められる。この値については、「*SNA Management Services*」を 2 照）、または 8 バイトのタイプ 1134 の EBCDIC 導入定義名のいずれかになります。この値は、フォーカル・ポイントが取り消され、まだ置換されていない場合（したがって、現 ~ 点でアクティブなフォーカル・ポイントがない場合）は、すべてゼロになります。

**fp_type**
フォーカル・ポイントのタイプ。詳細については、「*SNA Management Services*」を 2 照してください。

AP_EXPLICIT_PRIMARY_FP
AP_BACKUP_FP
AP_DEFAULT_PRIMARY_FP
AP_DOMAIN_FP
AP_HOST_FP
AP_NO_FP

**fp_status**
フォーカル・ポイントの状態。

**AP_NOT_ACTIVE**
フォーカル・ポイントは、アクティブから非アクティブになりました。

**AP_ACTIVE**
フォーカル・ポイントは、非アクティブまたはアクティブ保留からアクティブになりました。

**fp_routing**
アプリケーションが MDS トランスポートを H 用してフォーカル・ポイントにデータを送る ~ に X 定する経路 X 定のタイプ。この値は、フォーカル・ポイントの状態が AP_ACTIVE の場合にのみ有効です。

**AP_DEFAULT**
省略 ~ の経路 X 定を H って、フォーカル・ポイントに MDS_MU を送ります。

**AP_DIRECT**
MDS_MU はセッションを介して直接フォーカル・ポイントに経路 X 定されます。

# ISR_INDICATION

この verb はCommunications Serverにのみ適用します。

このX示は、ISR セッションがh動化または非h動化された~に生成されます。セッションが非h動化されると、最後のセッション統計が戻されます。セッションがh動化されると、 **pri_sess_stats** および **sec_sess_stats** フィールドは予約済みになります。

## VCB 構造体

```
typedef struct isr_indication
{
         unsigned short  opcode;             /* verb operation code            */
         unsigned char   reserv2;            /* reserved                       */
         unsigned char   format;             /* format                         */
         unsigned short  primary_rc;         /* primary return code            */
         unsigned long   secondary_rc;       /* secondary return code          */
         unsigned char   data_lost;          /* previous indication lost       */
         unsigned char   deactivated;        /* has ISR session been           */
                                             /* deactivated?                   */
         FQPCID          fqpcid;             /* fully qualified procedure      */
                                             /* correlator ID                  */
         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          */
         unsigned long   sense_data;         /* sense data                     */
         unsigned char   reserv2a[2];        /* reserved                       */
         SESSION_STATS   pri_sess_stats;     /* primary hop session stats      */
         SESSION_STATS   sec_sess_stats;     /* secondary hop session          */
                                             /* statistics                     */
         unsigned char   reserva[20];        /* reserved                       */
} ISR_INDICATION;

typedef struct fqpcid
{
        unsigned char   pcid[8];             /* pro correlator identifier      */
        unsigned char   fqcp_name[17];       /* orig's network qualified       */
                                             /* CP name                        */
        unsigned char   reserve3[3];         /* reserved                       */
} FQPCID;

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size        */
        unsigned short  send_ru_size;        /* session send RU size           */
        unsigned short  max_send_btu_size;   /* Maximum send BTU size          */
        unsigned short  max_rcv_btu_size;    /* Maximum rcv BTU size           */
        unsigned short  max_send_pac_win;    /* Max send pacing window size    */
        unsigned short  cur_send_pac_win;    /* Curr send pacing window size   */
        unsigned short  max_rcv_pac_win;     /* Max receive pacing win size    */
        unsigned short  cur_rcv_pac_win;     /* Curr rec pacing window size    */
        unsigned long   send_data_frames;    /* Number of data frames sent     */
        unsigned long   send_fmd_data_frames;/* num of FMD data frames sent    */
        unsigned long   send_data_bytes;     /* Number of data bytes sent      */
        unsigned long   rcv_data_frames;     /* Num data frames received       */
        unsigned long   rcv_fmd_data_frames; /* num of FMD data frames recvd   */
        unsigned long   rcv_data_bytes;      /* Num 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[8];         /* Link station name             */
        unsigned char   pacing_type;        /* type of pacing in use         */
} SESSION_STATS;
```

## Q i a ー ? ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_ISR_INDICATION

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**primary_rc**
AP_OK

**data_lost**
データが失われたかどうかを示します（AP_YES または AP_NO)。 この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。 アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**deactivate**
ISR セッションが非 h 動化された場合は AP_YES に設定され、セッションが h 動化された場合は AP_NO に設定されます。

**fqpcid.pcid**
プロシージャー相関関係R ID。 これは 8 バイトの 16 進数ストリングです。

**fqpcid.pcid_name**
完全修飾制御点名。この名前は、17 バイトの長さで、EBCDIC スペースを右に埋め込みます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**fqplu_name**
1 次 LU の完全修飾名（BIND 要求でX定)。この名前は、17 バイトの長さで、EBCDIC スペースを右に埋め込みます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）**deactivated** が AP_YES になっている場合、この名前はすべてゼロになります。

**fqslu_name**
2 次 LU の完全修飾名（BIND 要求でX定)。この名前は、17 バイトの長さで、EBCDIC スペースを右に埋め込みます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）**deactivated** が AP_YES になっている場合、この名前はすべてゼロになります。

cos_name<s_name&s_name8s_name"s_nameEBCDICs_name<s_name3Ns_name<s_name

**pri_sess_stats.sidl**
セッション ID の下位バイト。

**pri_sess_stats.odai**
原点宛先アドレスX示。セッション + O ~ に、BIND の送信側は、ローカル・ノードに 1 次リンク・ステーションがある場合、このフィールドをゼロに設定します。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には、このフィールドは 1 に設定されます。

**pri_sess_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。このフィールドをH用して、セッション統計を、セッション・トラフィックが流れるリンクと結びUけることができます。

**pri_sess_stats.pacing_type**
1 次セッションでH用中の受信ペーシングのタイプ。これは値 AP_NONE、AP_PACING_FIXED、または AP_PACING_ADAPTIVE を取れます。

**sec_sess_stats.rcv_ru_size**
最大の受信 RU サイズ。

**sec_sess_stats.send_ru_size**
最大の送信 RU サイズ。

**sec_sess_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**sec_sess_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**sec_sess_stats.max_send_pac_win**
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。

**sec_sess_stats.cur_send_pac_win**
このセッションの送信ペーシング・ウィンドウの現行サイズ。

**sec_sess_stats.max_rcv_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。

**sec_sess_stats.cur_rcv_pac_win**
このセッションの受信ペーシング・ウィンドウの現行サイズ。

**sec_sess_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**sec_sess_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**sec_sess_stats.send_data_bytes**
送信された通常フロー・データ・バイト数。

**sec_sess_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**sec_sess_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**sec_sess_stats.rcv_data_bytes**
受信された通常フロー・データ・バイト数。

**sec_sess_stats.sidh**
セッション ID 上位バイト。

**sec_sess_stats.sidl**
セッション ID の下位バイト。

**sec_sess_stats.odai**
原点宛先アドレスX示。セッション + O ~ に、BIND の送信側は、ローカル・ノードに 1 次リンク・ステーションがある場合、このフィールドをゼロに設定します。 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には、このフィールドは 1 に設定されます。

**sec_sess_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。このフィールドをH用して、セッション統計を、セッション・トラフィックが流れるリンクと結びUけることができます。

**sec_sess_stats.pacing_type**
2 次セッションでH用中の受信ペーシングのタイプ。これは値 AP_NONE、AP_PACING_FIXED、または AP_PACING_ADAPTIVE を取れます。

# LOCAL_LU_INDICATION

この X 示は、ローカル LU が定義または削除された ~ に生成されます。この X 示によって、登録済みのアプリケーションは、現 ~ 点で定義済みになっているすべてのローカル LU のリストを保 } できます。

## VCB 構造体

```
typedef struct local_lu_indication
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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   description[RD_LEN]; /* 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
{
        unsigned short  rcv_ru_size;         /* session receive RU size       */
        unsigned short  send_ru_size;        /* session send RU size          */
        unsigned short  max_send_btu_size;   /* max send BTU size             */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size              */
        unsigned short  max_send_pac_win;    /* max send pacing window size   */
        unsigned short  cur_send_pac_win;    /* current send pacing win size  */
        unsigned short  max_rcv_pac_win;     /* max receive pacing win size   */
        unsigned short  cur_rcv_pac_win;     /* curr receive pacing winsize   */
        unsigned long   send_data_frames;    /* number of data frames sent    */
        unsigned long   send_fmd_data_frames;/* num of FMD data frames sent   */
        unsigned long   send_data_bytes;     /* number of data bytes sent     */
        unsigned long   rcv_data_frames;     /* num of data frames received   */
        unsigned long   rcv_fmd_data_frames; /* num FMD data frames received  */
        unsigned long   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[8];          /* Link station name             */
        unsigned char   pacing_type;         /* type of pacing in use         */
} SESSION_STATS;
```

**注:** LU-SSCP 統計は、**nau_address** がゼロ以 0 の値になっていて、かつ LU-SSCP セッションがアクティブから非アクティブになった ~ にのみ有効です。 それ以 0 の場合は、フィールドが予約済みになります。

## Ｑｉａー？ー

**opcode**

AP_LOCAL_LU_INDICATION

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**
AP_OK

**secondary_rc**
ゼロになります。

**data_lost**
データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**reason**
X示が発行された理由。

**AP_ADDED**
LU が定義されました。

**AP_REMOVED**
LU が DELETE_LOCAL_LU によって明示的に削除されたか、DELETE_LS、DELETE_PORT、または DELETE_DLC によって暗黙に削除されました。

**AP_SSCP_ACTIVE**
ノードが ACTLU を正常に処理した後、LU-SSCP がアクティブになりました。

**AP_SSCP_INACTIVE**
DACTLU の正常な処理が行われた後、またはリンクの障 2 が発生した後に、 LU-SSCP セッションが非アクティブになりました。

**lu_name**
LU 名。状態が変化したローカル LU の名前。これは 8 バイトの英数 z タイプ A の EBCDIC 文 z 列（先頭は英 z ）で、右側の余白には EBCDIC スペースを埋め込みます。

**description**
リソース記述（DEFINE_LOCAL_LU でX定）。

**lu_alias**
ローカルに定義された LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてに意味があります。

**nau_address**
LU のネットワーク・アドレス可能単位のアドレス。0 ～ 255 の範囲になります。 ゼロ以 0 の値の場合、この LU は従属型 LU であり、ゼロの値はこの LU が独立 LU であることを示します。

**pu_name**
この LU がH用する PU の名前。これは、8 バイト英数 z のタイプ A の EBCDIC 文 z 列です。 このフィールドは、LU が従属型 LU（つまり、

**nau_address** がゼロ以 0 の値）である場合にのみ有効で、独立型 LU の場合にはすべて 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であるかどうかを示します（AP_YES または AP_NO）。 **nau_address** がゼロの場合、このフィールドは予約済みになります。

**lu_sscp_stats.rcv_ru_size**
このフィールドは常に予約済みです。

**lu_sscp_stats.send_ru_size**
このフィールドは常に予約済みです。

**lu_sscp_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**lu_sscp_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**lu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.cur_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.send_data_frames**
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_frames**
送信された通常フロー FMD データ・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・バイト数。

**lu_sscp_stats.rcv_data_frames**
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_frames**
受信された通常フロー FMD データ・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・バイト数。

**lu_sscp_stats.sidh**
セッション ID 上位バイト。

**lu_sscp_stats.sidl**
セッション ID の下位バイト。

**lu_sscp_stats.odai**
原点宛先アドレス X 示。セッション + O ~ に、 ACTLU の送信側は、ローカ

ル・ノードに 1 次リンク・ステーションがある場合、このフィールドにゼロを設定し、 ACTLU の送信側が 2 次リンク・ステーションのあるノードである場合は、1 を設定します。

**lu_sscp_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。このフィールドをH用して、このセッションとこのセッションが流れるリンクとを相関させることができます。

**lu_sscp_stats.pacing_type**
LU-SSCP セッションでH用中の受信ペーシングのタイプ。これは値 AP_NONE をとります。

**sscp_id**
これは、6 バイトのフィールドで、この LU がH用する PU 用の ACTPU に受信された SSCP ID を含みます。

このフィールドは、従属型 LU によってのみH用され、独立型 LU の場合、または **lu_sscp_sess_active** が AP_YES にセットされない場合には、すべて 2 進ゼロに設定されます。

# LOCAL_TOPOLOGY_INDICATION

このX示は、ノードのローカル・トポロジー・データベース内の TG 項目がアクティブから非アクティブに、または非アクティブからアクティブに変わった~に生成されます。

## VCB 構造体

```
typedef struct local_topology_indication
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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 session is active       */
        unsigned char   branch_link_type;    /* branch link type              */
        unsigned char   branch_tg;           /* TG is a branch TG             */
        unsigned char   reserva[17];         /* reserved                      */
} LOCAL_TOPOLOGY_INDICATION;
```

## Ｑｉａ－？－

**opcode**
AP_LOCAL_TOPOLOGY_INDICATION

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**
AP_OK

**secondary_rc**
ゼロになります。

**data_lost**
データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**status** TG の状況を示します。これは、以下の 1 つまたは複数の値を互いに OR 結合したものです。

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_NONE

**dest** TG の完全修飾宛先ノード名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つ

のタイプ A の EBCDIC 文 z の文 z 列で構成されます。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**dest_type**
ノードのタイプ。以下のいずれかの値になります。

AP_END_NODE
AP_NETWORK_NODE
AP_VRN

**tg_num**
TG と関連した番号。

**cp_cp_session_active**
ローカル・ノードの競合勝者 CP-CP セッションがアクティブ (AP_NO または AP_YES) であるかどうかを X 定します。

**branch_link_type**
BrNN のみ。この TG のこのブランチ・リンク・タイプ。これは、以下の 1 つに設定されます。

**AP_UPLINK**
このリンクはアップリンクです。

**AP_DOWNLINK**
このリンクは EN へのダウンリンクです。

**AP_DOWNLINK_TO_BRNN**
TG は EN フェイスを = 示している BrNN へのダウンリンクです。

**AP_OTHERLINK**
このリンクはアザーリンクです。

他のノード・タイプ : このフィールドは有意味ではなく、常に AP_BRNN_NOT_SUPPORTED がセットされています。

**branch_tg**
NN のみ。TG がブランチ TG であるかどうかを X 定します。

**AP_NO**
TG はブランチ TG ではありません。

**AP_YES**
TG はブランチ TG です。

他のノード・タイプ : このフィールドは有意味ではなく、常に AP_NO がセットされています。

# LS_INDICATION

このX示は、リンクをH用しているアクティブ・セッションの数が変わった場合、またはリンク・ステーションの0 t 状態が変更された場合に生成されます。リンク・ステーションが非アクティブになると、リンク・ステーションの統計が提供されます。

## VCB 構造体

```
typedef struct ls_indication
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* verb attributes               */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   secondary_rc;        /* secondary return code         */
        unsigned char   data_lost;           /* previous indication lost      */
        unsigned char   deactivated;         /* has session been deactivated? */
        unsigned char   ls_name[8];          /* link station name             */
        unsigned char   description[RD_LEN]; /* resource description          */
        unsigned char   adj_cp_name[17];     /* network qualified Adj CP name */
        unsigned char   adj_node_type;       /* adjacent node type            */
        unsigned short  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                     */
        unsigned long   sense_data;          /* sense data                    */
        unsigned char   brnn_link_type;      /* branch link type              */
        unsigned char   adj_cp_is_brnn;      /* adjacent CP is a BrNN         */
        unsigned char   reserva[17];         /* reserved                      */
} LS_INDICATION;

typedef struct ls_stats
{
        unsigned long   in_xid_bytes;        /* num of XID bytes received     */
        unsigned long   in_msg_bytes;        /* num message bytes received    */
        unsigned long   in_xid_frames;       /* num XID frames received       */
        unsigned long   in_msg_frames;       /* num message frames received   */
        unsigned long   out_xid_bytes;       /* num XID bytes sent            */
        unsigned long   out_msg_bytes;       /* num message bytes sent        */
        unsigned long   out_xid_frames;      /* number of XID frames sent     */
        unsigned long   out_msg_frames;      /* num message frames sent       */
        unsigned long   in_invalid_sna_frames;
                                             /* num invalid frames recvd      */
        unsigned long   in_session_control_frames;
                                             /* number of control             */
                                             /* frames recvd                  */
        unsigned long   out_session_control_frames;
                                             /* number of control             */
                                             /* frames sent                   */
        unsigned long   echo_rsps;           /* response from adj LS count    */
        unsigned long   current_delay;       /* time taken for last           */
                                             /* test signal                   */
        unsigned long   max_delay;           /* max delay by test signal      */
        unsigned long   min_delay;           /* min delay by test signal      */
        unsigned long   max_delay_time;      /* time since longest delay      */
        unsigned long   good_xids;           /* successful XID on LS count    */
        unsigned long   bad_xids;            /* unsuccessful XID on LS count  */
} LS_STATS;
```

## Q i a ー ? ー

**opcode**

AP_LS_INDICATION

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可 k 性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前の X 示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**deactivated**

LS が非アクティブになった場合は AP_YES に設定され、LS がアクティブになった場合は AP_NO に設定されます。

**ls_name**

リンク・ステーションの名前。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**description**

リソースの記述（DEFINE_LS で X 定）。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**adj_cp_name**

隣接 CP の 17 バイト完全修飾名。2 つのタイプ A の EBCDIC 文 z 列を EBCDIC のドットで区切った形になり、右側の余白は EBCDIC のスペースで埋められます。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**adj_node_type**

ノードのタイプ。以下のいずれかの値になります。

AP_END_NODE
AP_NETWORK_NODE
AP_LEN_NODE
AP_VRN

## LS_INDICATION

**act_sess_count**
リンクをH用しているアクティブ・セッションの合計数（エンドポイントおよび中間の両方）。

**indication_cause**
X示が生成された原因。以下のいずれかの値になります。

**AP_ACTIVATION_STARTED**
リンクのh動化が行われています。

**AP_ACTIVATING**
リンクがアクティブになりました。

**AP_DEACTIVATION_STARTED**
リンクの非h動化が行われています。

**AP_DEACTIVATING**
リンクが非アクティブになりました。

**AP_SESS_COUNT_CHANGING**
リンクをH用しているアクティブ・セッションの数が変わりました。

**AP_CP_NAME_CHANGING**
隣接ノードの CP 名が変更されました。

**AP_FAILED**
リンクに障2が発生しました。

**AP_ACTIVATION_FAILED**
リンクのh動化が失敗しました。

**AP_PENDING_RETRY**
再n行タイマーが+Oされました。タイマーが満了したときには、リンクのh動化は自動的に再n行されます。

**AP_DATA_LOST**
直前のX示が失われました。リンク・ステーションの統計が=示されるのは、リンク・ステーションがアクティブから非アクティブになった場合（つまり、**deactivated** に AP_YES が設定され、**indication_cause** が AP_DEACTIVATING に設定された場合）だけです。それ以0の場合は、フィールドが予約済みになります。

**ls_stats.in_xid_bytes**
このリンク・ステーション上で受信された XID（交換識別）の合計バイト数。

**ls_stats.in_msg_bytes**
このリンク・ステーション上で受信されたデータの合計バイト数。

**ls_stats.in_xid_frames**
このリンク・ステーション上で受信された XID（交換識別）フレームの合計数。

**ls_stats.in_msg_frames**
このリンク・ステーション上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・ステーション上で送信された XID（交換識別）の合計バイト数。

**ls_stats.out_msg_bytes**
このリンク・ステーション上で送信されたデータの合計バイト数。

**ls_stats.out_xid_frames**
このリンク・ステーション上で送信された XID（交換識別）フレームの合計数。

**ls_stats.out_msg_frames**
このリンク・ステーション上で送信されたデータ・フレームの合計数。

**ls_stats.in_invalid_sna_frames**
このリンク・ステーション上で受信された誤った SNA フレームの合計数。

**ls_stats.in_session_control_frames**
このリンク・ステーション上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_frames**
このリンク・ステーション上で送信されたセッション制御フレームの合計数。

**ls_stats.echo_rsps**
隣接ノードから受信されたエコー応答の数。エコー要求は、伝搬遅延を測定するために隣接ノードに対して周期的に送信されます。

**ls_stats.current_delay**
このリンク・ステーションから隣接リンク・ステーションに最後のテスト信号が送信されて戻ってくるまでの～間（ミリC）。

**ls_stats.max_delay**
このリンク・ステーションから隣接リンク・ステーションにテスト信号が送信されて戻ってくるまでの最長～間（ミリC）。

**ls_stats.min_delay**
このリンク・ステーションから隣接リンク・ステーションにテスト信号が送信されて戻ってくるまでの最短～間（ミリC）。

**ls_stats.max_delay_time**
システム起動～から最長遅延が発生した～までの～間（1/100 C単位）。

**ls_stats.good_xids**
このリンク・ステーションのO動～からこのリンク・ステーション上で成功した XID 交換の合計数。

**ls_stats.bad_xids**
このリンク・ステーションのO動～からこのリンク・ステーション上で失敗した XID 交換の合計数。

**tg_num**
TG と関連した番号。

**sense_data**
このセンス・データは、パーソナル・コミュニケーションズまたは Communications Serverが XID プロトコル・エラーを検出した～に設定されます。**indication_cause** が AP_FAILED になっている場合以0は、このフィールドは予約済みになります。

## LS_INDICATION

**brnn_link_type**

BrNN のみ。このブランチのリンク・タイプ。以下のいずれかになります。

**AP_UPLINK**

このリンクはアップリンクです。

**AP_DOWNLINK**

このリンクはダウンリンクです。

**AP_OTHERLINK**

このリンクはアザーリンクです。

**AP_UNKNOWN_LINK_TYPE**

このリンクはアザーリンクです。

他のノード・タイプ : このフィールドは有意味ではなく、常に AP_BRNN_NOT_SUPPORTED がセットされています。

**adj_cp_is_brnnt**

すべてのノード・タイプ: 隣接ノードが BrNN であるかどうかを X 定します。

**AP_UNKNOWN**

隣接ノードが BrNN であるかどうかは T 明です。

**AP_NO**

隣接ノードは BrNN ではありません。

**AP_YES**

隣接ノードは BrNN です。

# LU_0_TO_3_INDICATION

このX示は、ローカル LU（タイプ 0 ～ 3）の状態が変化した~に生成されます。

## VCB 構造体

```
typedef struct lu_0_to_3_indication
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   attributes;          /* attributes                    */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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[RD_LEN]; /* resource description          */
        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-SLU session statistics    */
        unsigned char   sscp_id[16];         /* SSCP ID                       */
} LU_0_TO_3_INDICATION;

typedef struct session_stats
{
        unsigned short  rcv_ru_size;         /* session receive RU size       */
        unsigned short  send_ru_size;        /* session send RU size          */
        unsigned short  max_send_btu_size;   /* max send BTU size             */
        unsigned short  max_rcv_btu_size;    /* max rcv BTU size              */
        unsigned short  max_send_pac_win;    /* max send pacing window size   */
        unsigned short  cur_send_pac_win;    /* current send pacing win size  */
        unsigned short  max_rcv_pac_win;     /* max receive pacing win size   */
        unsigned short  cur_rcv_pac_win;     /* curr receive pacing winsize   */
        unsigned long   send_data_frames;    /* number of data frames sent    */
        unsigned long   send_fmd_data_frames;
                                             /* num of FMD data frames sent   */
        unsigned long   send_data_bytes;     /* number of data bytes sent     */
        unsigned long   rcv_data_frames;     /* num of data frames received   */
        unsigned long   rcv_fmd_data_frames; /* num FMD data frames received  */
        unsigned long   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[8];          /* Link station name             */
        unsigned char   pacing_type;         /* type of pacing in use         */
} SESSION_STATS;
```

## Ｑｉａ－？－

**opcode**

AP_LU_0_TO_3_INDICATION

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

data_lost が AP_YES にセットされている場合には、
これは AP_EXTERNALLY_VISIBLE にセットされます。

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**
AP_OK

**secondary_rc**
ゼロになります。

**data_lost**
データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する，要があります。

**pu_name**
ローカル PU 名。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**lu_name**
状態が変化したローカル LU の名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**description**
リソースの記述（DEFINE_LU_0_TO_3 でX定）。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**nau_address**
LU のネットワーク・アドレス可能単位のアドレス。10 〜 2554 の範囲になります。

**lu_sscp_sess_active**
ACTLU が正常に処理されたかどうかを示します（AP_YES または AP_NO）。

**appl_conn_active**
アプリケーションがこの LU をH用している場合に設定されます（AP_YES または AP_NO）。

**plu_sess_active**
PLU-SLU セッションが h 動化されたかどうかを示します（AP_YES または AP_NO）。

**host_attachment**
LU ホスト処理装置接続機構のタイプを示します。

**AP_DLUR_ATTACHED**
LU は DLUR をH用して、ホスト・システムに接続しています。

**AP_DIRECT_ATTACHED**
LU は直接ホスト・システムと接続しています。LU-SSCP および

PLU-SLU の統計は、セッションがアクティブから非アクティブになった ~ にのみ有効です。それ以 0 の場合は、フィールドが予約済みになります。

**lu_sscp_stats.rcv_ru_size**
このフィールドは常に予約済みです。

**lu_sscp_stats.send_ru_size**
このフィールドは常に予約済みです。

**lu_sscp_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**lu_sscp_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**lu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.cur_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**lu_sscp_stats.send_data_frames**
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_frames**
送信された通常フロー FMD データ・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・バイト数。

**lu_sscp_stats.rcv_data_frames**
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_frames**
受信された通常フロー FMD データ・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・バイト数。

**lu_sscp_stats.sidh**
セッション ID 上位バイト。

**lu_sscp_stats.sidl**
セッション ID の下位バイト。

**lu_sscp_stats.odai**
原点宛先アドレス X 示。セッション + O ~ に、ACTLU の送信側は、ローカル・ノードに 1 次リンク・ステーションがある場合、このフィールドにゼロを設定し、ACTLU の送信側が 2 次リンク・ステーションのあるノードである場合は、1 を設定します。

**lu_sscp_stats.ls_name**
統計に関連 U けられているリンク・ステーション名。これは、ローカル = 示

可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。このフィールドをH用して、このセッションとこのセッションが流れるリンクとを相関させることができます。

**lu_sscp_stats.pacing_type**
LU-SSCP セッションでH用中の受信ペーシングのタイプ。これは値 AP_NONE をとります。

**plu_stats.rcv_ru_size**
最大の受信 RU サイズ。

**plu_stats.send_ru_size**
最大の送信 RU サイズ。

**plu_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**plu_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**plu_stats.max_send_pac_win**
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。

**plu_stats.cur_send_pac_win**
このセッションの送信ペーシング・ウィンドウの現行サイズ。

**plu_stats.max_rcv_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。

**plu_stats.cur_rcv_pac_win**
このセッションの受信ペーシング・ウィンドウの現行サイズ。

**plu_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**plu_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**plu_stats.send_data_bytes**
送信された通常フロー・データ・バイト数。

**plu_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**plu_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**plu_stats.rcv_data_bytes**
受信された通常フロー・データ・バイト数。

**plu_stats.sidh**
セッション ID 上位バイト。

**plu_stats.sidl**
セッション ID の下位バイト。

**plu_stats.odai**
原点宛先アドレスX示。セッション + O ~ に、ACTLU の送信側は、ローカ

ル・ノードに 1 次リンク・ステーションがある場合、このフィールドにゼロを設定し、ACTLU の送信側が 2 次リンク・ステーションのあるノードである場合は、1 を設定します。

**plu_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。このフィールドをH用して、このセッションとこのセッションが流れるリンクとを相関させることができます。

**plu_stats.pacing_type**
PLU-SLU セッションでH用中の受信ペーシングのタイプ。これは値 AP_NONE または AP_PACING_FIXED を取ります。

**sscp_id**
これは、6 バイトのフィールドで、この LU がH用する PU 用の ACTPU に受信された SSCP ID を含みます。

**lu_sscp_sess_active** が AP_YES でない場合には、このフィールドはゼロとなります。

# MODE_INDICATION

このX示は、ローカル LU とパートナー LU の組み合わせが特定のモードをH用しはじめた~、およびローカル LU とパートナー LU とモードの組み合わせによる現行セッションのカウントが変化した~に送信されます。

## VCB 構造体

```
typedef struct mode_indication
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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[RD_LEN]; /* resource description          */
        unsigned short  curr_sess_count;     /* current session count         */
        unsigned char   reserva[20];         /* reserved                      */
} MODE_INDICATION;
```

## Ｑｉａ－？－

**opcode**

AP_MODE_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する，要があります。

**removed**

項目が除去されるかどうかを示します（AP_YES または AP_NO）。この値は、項目が追加される~ではなく除去される~に設定されます。

**lu_alias**

ローカルに定義された LU の別名。これは、ローカル=示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**plu_alias**

パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**fqplu_name**

パートナー LU の 17 バイト完全修飾ネットワーク名。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式で X 定します。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**mode_name**

セッション・グループのネットワーク特性を X 定するモード名。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列 (先頭は文 z ) です。EBCDIC スペースが右の余白に埋め込まれます。

**description**

リソースの記述 (DEFINE_MODE で X 定)。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**curr_sess_count**

ローカル LU とパートナー LU とモードの組み合わせによるセッションの現行カウント。

# NN_TOPOLOGY_NODE_INDICATION

この verb はCommunications Serverにのみ適用します。

このX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アクティブから非アクティブに、または非アクティブからアクティブになった~に生成されます。

## VCB 構造体

```
typedef struct nn_topology_node_indication
{
        unsigned short  opcode;             /* verb operation code            */
        unsigned char   reserv2;            /* reserved                       */
        unsigned char   format;             /* format                         */
        unsigned short  primary_rc;         /* primary return code            */
        unsigned long   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;       /* node is branch aware           */
        unsigned char   reserva[19];        /* reserved                       */
} NN_TOPOLOGY_NODE_INDICATION;
```

## Ｑｉａー？ー

**opcode**

AP_NN_TOPOLOGY_TG_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO)。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する，要があります。

**deactivated**

ノードが非アクティブになった場合は AP_YES に設定され、ノードがアクティブになった場合は AP_NO に設定されます。

**node_name**

ネットワーク・トポロジー・データベースから得られたネットワーク修飾ノード名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。 EBCDIC ドットによって連結される2 つのタイプ A の EBCDIC 文z の文z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**node_type**
ノードのタイプ。以下のいずれかの値になります。

AP_NETWORK_NODE
AP_VRN

**branch_aware**
ノードがブランチを認識しているかどうかをX定します。

**AP_NO**
ノードはブランチを認識していない。

**AP_YES**
ノードはブランチを認識している。

# NN_TOPOLOGY_TG_INDICATION

この verb はCommunications Serverにのみ適用します。

このX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 TG 項目がアクティブから非アクティブに、または非アクティブからアクティブになった~に生成されます。

## VCB 構造体

```
typedef struct nn_topology_tg_indication
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   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 the TG */
        unsigned char   dest_type;          /* TG destination node type      */
        unsigned char   cp_cp_session_active;
                                            /* CP-CP session is active       */
        unsigned char   branch_tg;          /* TG is a branch TG             */
        unsigned char   reserva[16];        /* reserved                      */
} NN_TOPOLOGY_TG_INDICATION;
```

## Ｑｉａ－？－

**opcode**

AP_NN_TOPOLOGY_TG_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。**data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する，要があります。

**status** TG の状況を示します。これは、以下の 1 つまたは複数を互いに OR 結合したもです。

AP_TG_OPERATIVE
AP_TG_QUIESCING
AP_TG_CP_CP_SESSIONS
AP_NONE

**owner** TG の発信側ノードの名前（ , ずローカル・ノード名に設定されます)。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**dest** TG の完全修飾宛先ノード名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**tg_num**
TG と関連した番号。

**owner_type**
TG を所有するノードのタイプ。

AP_NETWORK_NODE
AP_VRN

**dest_type**
ノードのタイプ。

AP_NETWORK_NODE
AP_VRN

**cp_cp_session_active**
所有する競合勝者 CP-CP セッションがアクティブ (AP_NO または AP_YES) であるかどうかを X 定します。

**branch_tg**
TG がブランチ TG であるかどうかを X 定します。

**AP_NO**
TG はブランチ TG ではありません。

**AP_YES**
TG はブランチ TG です。

# PLU_INDICATION

このX示は、ローカル LU がパートナー LU と最初に接続した~に生成されます。これはつまり、この PLU に対する最初の ALLOCATE が処理された~、またはこの PLU から最初の BIND が送られてきた~になります。このX示は、パートナー CP の名前が変更された~にも生成されます。

## VCB 構造体

```
typedef struct plu_indication
{
        unsigned short  opcode;              /* verb operation code           */
        unsigned char   reserv2;             /* reserved                      */
        unsigned char   format;              /* format                        */
        unsigned short  primary_rc;          /* primary return code           */
        unsigned long   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[RD_LEN]; /* resource description          */
        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;
```

## Ｑｉａー？ー

**opcode**

AP_PLU_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

1 つまたは複数のX示が失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が直前のX示を送信できなかった~に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**removed**

項目が除去されるかどうかを示します（AP_YES または AP_NO）。この値は、項目が追加される~ではなく除去される~に設定されます。

**lu_alias**

ローカルに定義された LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**plu_alias**

パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**fqplu_name**

パートナー LU の 17 バイト完全修飾ネットワーク名。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式で X 定します。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**description**

リソースの記述 (DEFINE_PARTNER_LU で X 定)。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**partner_cp_name**

パートナー LU の CP の 17 バイト完全修飾ネットワーク名。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式で X 定します。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**partner_lu_located**

パートナー CP の名前が解決されたかどうか、つまり **partner_cp_name** フィールドにその CP 名があるかどうかを示します (AP_YES または AP_NO)。

# PORT_INDICATION

このX示は、ポートがアクティブから非アクティブに（またはその逆に）なった~に生成されます。

## VCB 構造体

```
typedef struct port_indication
{
   unsigned short  opcode;              /* verb operation code           */
   unsigned char   reserv2;             /* reserved                      */
   unsigned char   attributes;          /* verb attributes               */
   unsigned char   format;              /* format                        */
   unsigned short  primary_rc;          /* primary return code           */
   unsigned long   secondary_rc;        /* secondary return code         */
   unsigned char   data_lost;           /* previous indication lost      */
   unsigned char   deactivated;         /* has session been deactivated? */
   unsigned char   port_name[8];        /* link station name             */
   unsigned char   description[RD_LEN]; /* resource description          */
   unsigned char   reserva[20];         /* reserved                      */
} PORT_INDICATION;
```

## Ｑｉａ－？－

**opcode**

AP_PORT_INDICATION

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**deactivated**

ポートが非アクティブになった場合は AP_YES に設定され、ポートがアクティブになった場合は AP_NO に設定されます。

**port_name**

ポート名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**description**

リソースの記述（DEFINE_PORT で X 定）。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

# PU_INDICATION

このX示は、ローカル PU の状態が変化した~に生成されます。

## VCB 構造体

```
typedef struct pu_indication
{
   unsigned short  opcode;               /* verb operation code           */
   unsigned char   attributes;           /* attributes                    */
   unsigned char   reserv2;              /* reserved                      */
   unsigned char   format;               /* format                        */
   unsigned short  primary_rc;           /* primary return code           */
   unsigned long   secondary_rc;         /* secondary return code         */
   unsigned char   data_lost;            /* previous indication lost      */
   unsigned char   pu_name[8];           /* PU Name                       */
   unsigned char   description[RD_LEN];  /* resource description          */
   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
{
   unsigned short  rcv_ru_size;          /* session receive RU size       */
   unsigned short  send_ru_size;         /* session send RU size          */
   unsigned short  max_send_btu_size;    /* max send BTU size             */
   unsigned short  max_rcv_btu_size;     /* max rcv BTU size              */
   unsigned short  max_send_pac_win;     /* max send pacing window size   */
   unsigned short  cur_send_pac_win;     /* curr send pacing window size  */
   unsigned short  max_rcv_pac_win;      /* max rcv pacing window size    */
   unsigned short  cur_rcv_pac_win;      /* curr receive pacing winsize   */
   unsigned long   send_data_frames;     /* number of data frames sent    */
   unsigned long   send_fmd_data_frames; /* num FMD data frames sent      */
   unsigned long   send_data_bytes;      /* number of data bytes sent     */
   unsigned long   rcv_data_frames;      /* num of data frames received   */
   unsigned long   rcv_fmd_data_frames;  /* num FMD data frames received  */
   unsigned long   rcv_data_bytes;       /* num 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[8];           /* Link station name             */
   unsigned char   pacing_type;          /* type of pacing in use         */
} SESSION_STATS;
```

## Ｑｉａ－？－

**opcode**

AP_PU_INDICATION

**attributes**

verb の属性。このフィールドはビット・フィールドです。最初のビットには定義されるべきリソースの可k性が含まれ、以下の 1 つに対応します。

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

data_lost が AP_YES にセットされている場合には、
これは AP_EXTERNALLY_VISIBLE にセットされます。

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**
AP_OK

**secondary_rc**
ゼロになります。

**data_lost**
データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**pu_name**
PU 名（DEFINE_LS verb で構成）。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列 (先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**description**
リソースの記述（DEFINE_LS または DEFINE_INTERNAL_PU でX定）。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**pu_sscp_sess_active**
ACTPU が正常に処理されたかどうかを示します（AP_YES または AP_NO）。

**host_attachment**
PU ホスト処理装置接続機構のタイプ。

**AP_DLUR_ATTACHED**
PU は DLUR をH用してホスト・システムと接続しています。

**AP_DIRECT_ATTACHED**
PU は直接ホスト・システムと接続しています。

**注:** PU-SSCP 統計は、セッションの状態がアクティブから非アクティブになった場合にのみ有効です。
それ以 0 の場合は、以下のフィールドが予約済みになります。

**pu_sscp_stats.rcv_ru_size**
このフィールドは常に予約済みです。

**pu_sscp_stats.send_ru_size**
このフィールドは常に予約済みです。

**pu_sscp_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**pu_sscp_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**pu_sscp_stats.max_send_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_sscp_stats.cur_send_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_sscp_stats.max_rcv_pac_win**
このフィールドは、常にゼロに設定されます。

**pu_sscp_stats.cur_rcv_pac_win**
このフィールドは、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_frames**
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_frames**
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・バイト数。

**pu_sscp_stats.sidh**
セッション ID の上位バイト。

**pu_sscp_stats.sidl**
セッション ID の下位バイト。

**pu_sscp_stats.odai**
原点宛先アドレス X 示。セッション + O ~ に、ACTPU の送信側は、ローカル・ノードに 1 次リンク・ステーションがある場合、このフィールドにゼロを設定し、ACTPU の送信側が 2 次リンク・ステーションのあるノードである場合は、1 を設定します。

**pu_sscp_stats.ls_name**
統計に関連 U けられているリンク・ステーション名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。このフィールドを H 用して、このセッションとこのセッションが流れるレル

# REGISTER_INDICATION_SINK

REGISTER_INDICATION_SINK はX示が送信されるプロセスと待ち行列をレジスターします。

REGISTER_INDICATION_SINK の verb_signal ヘッダーにある **orig_verb_data** は受信されたすべてのX示の verb_signal ヘッダー内に戻されます。

## VCB 構造体

```
typedef struct port_indication
{
   unsigned short  opcode;            /* verb operation code        */
   unsigned char   reserv2;           /* reserved                   */
   unsigned char   format;            /* format                     */
   unsigned short  primary_rc;        /* primary return code        */
   unsigned long   secondary_rc;      /* secondary return code      */
   unsigned PROC_ID   proc_id;        /* process identifier of sink */
   unsigned QUEUE_ID  queue_id;       /* queue identifier where     */
                                      /* indications will be sent   */
   unsigned short  indication_opcode; /* opcode of indication to    */
                                      /* be sunk                    */
} REGISTER_INDICATION_SINK;
```

## Ｑｉａー？ー

**opcode**

AP_REGISTER_INDICATION_SINK

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**proc_id**

受信プロセスのプロセス ID。

**queue_id**

受信プロセスのX示が送信される待ち行列 ID 。

**indication_opcode**

X示シンクが登録済みになると、生成されたときに常に戻されるX示のOpcode 。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_OP_CODE

AP_DYNAMIC_LOAD_ALREADY_REGD

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

関係する START_NODE パラメーターが送信されなかったために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_FUNCTION_NOT_SUPPORTED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

STOP_NODE verb が発行されていないために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# REGISTRATION_FAILURE

REGISTRATION_FAILURE は、ネットワーク・ノード・サーバーに q 源を登録しようとする n みが失敗した ~ に生成される X 示です。

## VCB 構造体

```
typedef struct registration_failure
{
   unsigned short  opcode;              /* verb operation code      */
   unsigned char   reserv2;             /* reserved                 */
   unsigned char   format;              /* format                   */
   unsigned short  primary_rc;          /* primary return code      */
   unsigned long   secondary_rc;        /* secondary return code    */
   unsigned char   data_lost;           /* previous indication lost */
   unsigned char   resource_name[17];   /* network qualified        */
                                        /* resource name            */
   unsigned short  resource_type;       /* resource type            */
   unsigned char   description[RD_LEN]; /* resource description     */
   unsigned char   reserv2b[2];         /* reserved                 */
   unsigned long   sense_data;          /* sense data               */
   unsigned char   reserva[20];         /* reserved                 */
} REGISTRATION_FAILURE;
```

## Ｑｉａ－？－

**opcode**

AP_REGISTRATION_FAILURE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンを X 定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO）。この値は、内 t 構成要素が、直前の X 示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**resource_name**

登録に失敗した q 源の名前。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。2 つのタイプ A の EBCDIC 文 z 列を EBCDIC のドットで区切った形になります。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**resource_type**

q 源タイプ。以下のいずれかの値になります。

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

# RTP_INDICATION

このX示は以下の場合に生成されます。

- RTP 接続が設定または切断された場合
- アクティブ・セッション・カウントが変わった場合
- 接続がパス・スイッチを実行した場合

接続が切断されると、最後の RTP 統計が戻されます。その他の場合は、**rtp_stats** フィールドが予約済みになります。

## VCB 構造体

```
typedef struct rtp_indication
{
   unsigned short  opcode;           /* verb operation code          */
   unsigned char   reserv2;          /* reserved                     */
   unsigned char   format;           /* format                       */
   unsigned short  primary_rc;       /* primary return code          */
   unsigned long   secondary_rc;     /* secondary return code        */
   unsigned char   data_lost;        /* previous indication(s) lost  */
   unsigned char   connection_state; /* the current state of the RTP */
                                     /* connection                   */
   unsigned char   rtp_name[8];      /* name of the RTP connection   */
   unsigned short  num_sess_active;  /* number of active sessions    */
   unsigned char   indication_cause; /* reason for this indication   */
   unsigned char   reserv3[3];       /* reserved                     */
   RTP_STATISTICS  rtp_stats;        /* RTP statistics               */
} RTP_INDICATION;

typedef struct rtp_statistics
{
   unsigned long   bytes_sent;         /* total num of bytes sent        */
   unsigned long   bytes_received;     /* total num bytes received       */
   unsigned long   bytes_resent;       /* total num of bytes resent      */
   unsigned long   bytes_discarded;    /* total num bytes discarded      */
   unsigned long   packets_sent;       /* total num of packets sent      */
   unsigned long   packets_received;   /* total num packets received     */
   unsigned long   packets_resent;     /* total num of packets resent    */
   unsigned long   packets_discarded;  /* total num packets discarded    */
   unsigned long   gaps_detected;      /* gaps detected                  */
   unsigned long   send_rate;          /* current send rate              */
   unsigned long   max_send_rate;      /* maximum send rate              */
   unsigned long   min_send_rate;      /* minimum send rate              */
   unsigned long   receive_rate;       /* current receive rate           */
   unsigned long   max_receive_rate;   /* maximum receive rate           */
   unsigned long   min_receive_rate;   /* minimum receive rate           */
   unsigned long   burst_size;         /* current burst size             */
   unsigned long   up_time;            /* total uptime of connection     */
   unsigned long   smooth_rtt;         /* smoothed round-trip time       */
   unsigned long   last_rtt;           /* last round-trip time           */
   unsigned long   short_req_timer;    /* SHORT_REQ timer duration       */
   unsigned long   short_req_timeouts; /* number of SHORT_REQ timeouts   */
   unsigned long   liveness_timeouts;  /* number of liveness timeouts    */
   unsigned long   in_invalid_sna_frames;
                                       /* number of invalid SNA frames   */
                                       /* received                       */
   unsigned long   in_sc_frames;       /* number of SC frames received   */
   unsigned long   out_sc_frames;      /* number of SC frames sent       */
   unsigned char   reserve[40];        /* reserved                       */
} RTP_STATISTICS;
```

## RTP_INDICATION

Ｑｉａ一？一

**opcode**

AP_RTP_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO)。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、直前のX示を受け取ってから複数回にわたりデータが変更された可能性があります。

**connection_state**

RTP 接続の現在の状態。以下のいずれかの値になります。

**AP_CONNECTING**

接続の設定が + O されましたが、まだ終了していません。

**AP_CONNECTED**

接続が完全にアクティブになっています。

**AP_DISCONNECTED**

接続はもはやアクティブではありません。

**rtp_name**

RTP 接続名。この名前は、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**num_sess_active**

接続上で現在アクティブなセッションの数。

**indication_cause**

X示が生成された原因。以下のいずれかの値になります。

**AP_ACTIVATED**

接続がアクティブになりました。

**AP_DEACTIVATED**

接続が非アクティブになりました。

**AP_PATH_SWITCHED**

接続がパス・スイッチを正常に終了しました。

**AP_SESS_COUNT_CHANGING**

接続をH用しているアクティブ・セッションの数が変わりました。

**AP_SETUP_FAILED**

接続が完全にアクティブになる前に失敗しました。RTP 接続統計は、接続が非アクティブになった ~ 、つまり **indication_cause** に

AP_DEACTIVATED または AP_SETUP_FAILED が設定された ~ にのみ = 示されます。それ以 0 の場合は、フィールドが予約済みになります。

**rtp_stats.bytes_sent**
ローカル・ノードがこの RTP 接続で送信した合計バイト数。

**rtp_stats.bytes_received**
ローカル・ノードがこの RTP 接続で受信した合計バイト数。

**rtp_stats.bytes_resent**
ローカル・ノードが転送中の消失のために再送信した合計バイト数。

**rtp_stats.bytes_discarded**
RTP 接続の相手方から送信されたものの、すでに受信したデータと重複していたために破棄された合計バイト数。

**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。

**rtp_stats.packets_received**
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**
ローカル・ノードが転送中の消失のために再送信したパケットの合計数。

**rtp_stats.packets_discarded**
RTP 接続の相手方から送信されたパケットの中で、すでに受信したデータと重複していたために破棄されたパケットの合計数。

**rtp_stats.gaps_detected**
ローカル・ノードが検出したギャップの合計数。それぞれのギャップは、1 つまたは複数の消失フレームに相当します。

**rtp_stats.send_rate**
この RTP 接続での現行送信速度（キロビット/C）。これは、ARB アルゴリズムによって計；される最高許容送信速度です。

**rtp_stats.max_send_rate**
この RTP 接続での最高送信速度（キロビット/C）。

**rtp_stats.min_send_rate**
この RTP 接続での最低送信速度（キロビット/C）。

**rtp_stats.receive_rate**
この RTP 接続での現行受信速度（キロビット/C）。これは、前回の測定 ~ 間帯に計；された実際の受信速度です。

**rtp_stats.max_receive_rate**
この RTP 接続での最高受信速度（キロビット/C）。

**rtp_stats.min_receive_rate**
この RTP 接続での最低受信速度（キロビット/C）。

**rtp_stats.burst_size**
RTP 接続での現在のバースト・サイズ（バイト）。

**rtp_stats.up_time**
RTP 接続がアクティブになっている合計 C 数。

## RTP_INDICATION

**rtp_stats.smooth_rtt**
ローカル・ノードとパートナー RTP ノードの間の平均往 | ~ 間（ミリ C）。

**rtp_stats.last_rtt**
ローカル・ノードとパートナー RTP ノードの間の最新往 | ~ 間（ミリ C）。

**rtp_stats.short_req_timer**
SHORT_REQ タイマーに設定している現在の ~ 間（ミリ C）。

**rtp_stats.short_req_timeouts**
この RTP 接続で SHORT_REQ タイマーが切れた合計回数。

**rtp_stats.liveness_timeouts**
この RTP 接続で h 性タイマーが切れた合計回数。 h 性タイマーは、接続が **rtp_connection_detail.liveness_timer** で X 定した ~ 間アイドルが続けば有効期限が切れになります。

**rtp_stats.in_invalid_sna_frames**
この RTP 接続で受信され、しかも無効であるとして破棄された 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。

# SESSION_INDICATION

このX示は、セッションがh動化または非h動化された~に生成されます。セッションが非h動化されると、最後のセッション統計が戻されます。セッションがh動化されると、**sess_stats** フィールドが予約済みになります。

## VCB 構造体

```
typedef struct session_indication
{
   unsigned short  opcode;              /* verb operation code           */
   unsigned char   reserv2;             /* reserved                      */
   unsigned char   format;              /* format                        */
   unsigned short  primary_rc;          /* primary return code           */
   unsigned long   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     */
   unsigned long   sense_data;          /* sense_data                    */
   unsigned char   duplex_support;      /* full-duplex support           */
   SESSION_STATS   sess_stats;          /* session statistics            */
   unsigned char   sscp_id[6];          /* SSCP ID of host               */
   unsigned char   plu_slu_comp_lvl;    /* PLU to SLU compression level  */
   unsigned char   slu_plu_comp_lvl;    /* SLU to PLU compression level  */
                                        /* correlator ID                 */
   unsigned char   reserva[12];         /* 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
{
   unsigned short  rcv_ru_size;         /* session receive RU size       */
   unsigned short  send_ru_size;        /* session send RU size          */
   unsigned short  max_send_btu_size;   /* max send BTU size             */
   unsigned short  max_rcv_btu_size;    /* max rcv BTU size              */
   unsigned short  max_send_pac_win;    /* max send pacing window size   */
   unsigned short  cur_send_pac_win;    /* curr send pacing window size  */
   unsigned short  max_rcv_pac_win;     /* max receive pacing win size   */
   unsigned short  cur_rcv_pac_win;     /* curr receive pacing winsize   */
   unsigned long   send_data_frames;    /* number of data frames sent    */
   unsigned long   send_fmd_data_frames; /* num FMD data frames sent     */
   unsigned long   send_data_bytes;     /* number of data bytes sent     */
   unsigned long   rcv_data_frames;     /* num data frames received      */
   unsigned long   rcv_fmd_data_frames; /* num FMD data frames received  */
   unsigned long   rcv_data_bytes;      /* num 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[8];          /* Link station name            */
   unsigned char   pacing_type;         /* type of pacing in use        */
   unsigned char   reserve;             /* reserved                     */
} SESSION_STATS;
```

## Q i a ー ? ー

**opcode**

AP_SESSION_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**secondary_rc**

ゼロになります。

**data_lost**

データが失われたかどうかを示します（AP_YES または AP_NO)。この値は、内 t 構成要素が、直前のX示が失われた障 2 を検出した場合に設定されます。 **data_lost** フラグに AP_YES が設定された場合、これ以降のデータ・フィールドには NULL が設定されることがあります。アプリケーションでは、失われた情報を更新するために照会 verb を発行する , 要があります。

**deactivated**

セッションが h 動化された場合は AP_NO に設定され、セッションが非 h 動化された場合は AP_YES に設定されます。

**lu_name**

LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。

**lu_alias**

ローカルに定義された LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてに意味があります。

**plu_alias**

パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。

**fqplu_name**

パートナー LU の 17 バイト完全修飾ネットワーク名。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式でX定します。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**mode_name**

セッション・グループのネットワーク特性をX定するモード名。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列 (先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**session_id**

8 バイトのセッション ID。

**fqpcid.pcid**
プロシージャー相関関係R ID。これは 8 バイトの 16 進数ストリングです。

**fqpcid.fqcp_name**
完全修飾 CP 名。この名前の長さは 17 バイトで、EBCDIC スペースが右の余白に埋め込まれます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文 z の文 z 列で構成されます。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**sense_data**
UNBIND 要求によって送信または受信されるセンス・データ。このフィールドは、**deactivated** が AP_NO の場合は予約済みになります。

**duplex_support**
BIND での交渉によって合意した会話二重サポートを戻します。以下のいずれかの値になります。

**AP_HALF_DUPLEX**
半二重会話だけがサポートされます。

**AP_FULL_DUPLEX**
半二重会話に加えて全二重会話もサポートされます。

**AP_UNKNOWN**
パートナー LU との間にアクティブ・セッションがないために、会話二重サポートは認識されません。

**sess_stats.rcv_ru_size**
最大の受信 RU サイズ。

**sess_stats.send_ru_size**
最大の送信 RU サイズ。

**sess_stats.max_send_btu_size**
送信可能な最大 BTU のサイズ。

**sess_stats.max_rcv_btu_size**
受信可能な最大 BTU のサイズ。

**sess_stats.max_send_pac_win**
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。

**sess_stats.cur_send_pac_win**
このセッションの送信ペーシング・ウィンドウの現行サイズ。

**sess_stats.max_rcv_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。

**sess_stats.cur_rcv_pac_win**
このセッションの受信ペーシング・ウィンドウの現行サイズ。

**sess_stats.send_data_frames**
送信された通常フロー・データ・フレームの数。

**sess_stats.send_fmd_data_frames**
送信された通常フロー FMD データ・フレームの数。

**sess_stats.send_data_bytes**
送信された通常フロー・データ・バイト数。

## SESSION_INDICATION

**sess_stats.rcv_data_frames**
受信された通常フロー・データ・フレームの数。

**sess_stats.rcv_fmd_data_frames**
受信された通常フロー FMD データ・フレームの数。

**sess_stats.rcv_data_bytes**
受信された通常フロー・データ・バイト数。

**sess_stats.sidh**
セッション ID の上位バイト。

**sess_stats.sidl**
セッション ID の下位バイト。

**sess_stats.odai**
原点宛先アドレスX示。セッションを立ち上げた~、ローカル・ノードに 1 次リンク・ステーションが入っている場合にはバインドの送信側はフィールドをゼロにセットします。また、バインドの送信側が 2 次リンク・ステーションが入ったノードの場合には、フィールドを 1 にセットします。

**sess_stats_stats.ls_name**
統計に関連Uけられているリンク・ステーション名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。8 バイトすべてに意味があります。このフィールドをH用して、セッション統計を、セッション・トラフィックが流れるリンクと結びUけることができます。

**sess_stats_pacing_type**
このセッションでH用中の受信ペーシングのタイプ。これは値 AP_PACING_ADAPTVE または AP_PACING_FIXED を取れます。

**sscp_id**
従属 LU の場合、このフィールドは、ローカル LU がマップされる PU のホストから ACTPU に受信された SSCP ID を含みます。独立型 LU の場合、このフィールドはすべて 2 進ゼロにセットされます。

**plu_slu_comp_lvl**
PLU から SLU に送信されるデータの圧縮レベルを示します。

**AP_NONE**
圧縮はH用されていません。

**AP_RLE_COMPRESSION**
RLE 圧縮がH用されています。

**AP_LZ9_COMPRESSION**
このノードは LZ9 圧縮をサポートできます。

**AP_LZ10_COMPRESSION**
ノードは LZ10 圧縮をサポートできます。

**AP_LZ12_COMPRESSION**
ノードは LZ12 圧縮をサポートできます。

**slu_plu_comp_lvl**
SLU から PLU に送信されるデータの圧縮レベルを示します。

**AP_NONE**
圧縮はH用されていません。

**AP_RLE_COMPRESSION**
RLE 圧縮がH用されています。

**AP_LZ9_COMPRESSION**
このノードは LZ9 圧縮をサポートできます。

**AP_LZ10_COMPRESSION**
ノードは LZ10 圧縮をサポートできます。

**AP_LZ12_COMPRESSION**
ノードは LZ12 圧縮をサポートできます。

# SESSION_FAILURE_INDICATION

このX示は、セッションが非h動化された~に生成されます。このX示は保証されます、すなわち、 , ず生成されます。

## VCB 構造体

```
typedef struct session_failure_indication
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   reserv3[3];      /* 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                 */
   unsigned long   sense_data;      /* sense_data                 */
  } SESSION_FAILURE_INDICATION;
```

## Ｑｉａ－？－

**opcode**

AP_SESSION_FAILURE_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**primary_rc**

AP_OK

**lu_name**

LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文z列です。

**lu_alias**

ローカルに定義された LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。8 バイトすべてに意味があります。

**plu_alias**

パートナー LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。

**fqplu_name**

パートナー LU の 17 バイト完全修飾ネットワーク名。この名前は、2 つのタイプ A の EBCDIC 文z列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式でX定します。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**mode_name**

セッション・グループのネットワーク特性をX定するモード名。これは、8 バイト、英数z、タイプ A の EBCDIC 文z列 (先頭は文z) です。EBCDIC スペースが右の余白に埋め込まれます。

**session_id**
8 バイトのセッション ID。

**sense_data**
セッション非 h 動化の原因を詳述したセンス・データ。

# UNREGISTER_INDICATION_SINK

UNREGISTER_INDICATION_SINK は、非送信請求X示を受信しているプロセスと待ち行列の識別を削除します。

**proc_id**、**queue_id** と **indication_opcode** のX定された組み合わせが一度でも登録されていると、項目は削除されます。X定された組み合わせが 2 度以上登録されている場合には、UNREGISTER_INDICATION_SINK の verb_signal 内の **orig_verb_data** にマッチングする項目は除去されます。

## VCB 構造体

```
typedef struct port_indication
{
   unsigned short  opcode;            /* verb operation code        */
   unsigned char   reserv2;           /* reserved                   */
   unsigned char   format;            /* format                     */
   unsigned short  primary_rc;        /* primary return code        */
   unsigned long   secondary_rc;      /* secondary return code      */
   unsigned PROC_ID   proc_id;        /* process identifier of sink */
   unsigned QUEUE_ID  queue_id;       /* queue identifier where     */
                                      /* indications will be sent   */
   unsigned short  indication_opcode; /* opcode of indication to    */
                                      /* be sunk                    */
} REGISTER_INDICATION_SINK;
```

## Ｑｉａー？ー

**opcode**

AP_UNREGISTER_INDICATION_SINK

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**proc_id**

X示が送信されているプロセスのプロセス ID。

**queue_id**

X示が送信されている待ち行列の、待ち行列 ID。

**indication_opcode**

戻されているX示の Opcode 。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_PARAMETER_CHECK

**secondary_rc**

AP_INVALID_OP_CODE

AP_DYNAMIC_LOAD_ALREADY_REGD

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**

AP_STATE_CHECK

**secondary_rc**

AP_INVALID_LU_NAME

関係する START_NODE パラメーターが送信されなかったために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_FUNCTION_NOT_SUPPORTED

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT_STARTED

STOP_NODE verb が発行されていないために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**

AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# 第10章 機密保護 verb

この章では、機密保護のためのパスワードの定義と削除にH用する verb について説明します。

# CONV_SECURITY_BYPASS

CONV_SECURITY_BYPASS は、「プログラム」 がローカル LU に会話レベルの機密保護を実 \ するかどうかを制御することをアプリケーションに許可します。一度機密保護がバイパスされてしまうと、「プログラム」はローカル LU での会話に認証または許可をしません。

## VCB 構造体

```
typedef struct conv_security_bypass
{
   unsigned short  opcode;           /* verb operation code        */
   unsigned char   reserv2;          /* reserved                   */
   unsigned char   format;           /* format                     */
   unsigned short  primary_rc;       /* primary return code        */
   unsigned long   secondary_rc;     /* secondary return code      */
   unsigned char   lu_name[8];       /* local LU name              */
   unsigned char   lu_alias[8];      /* local LU alias             */
   unsigned char   bypass_security;  /* should security be         */
                                     /* bypassed?                  */
   unsigned char   reserv3[3];       /* reserved                   */
} DEFINE_LU_LU_PASSWORD;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_CONV_SECURITY_BYPASS

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

ローカル LU の LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文 z 列です。このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、**lu_alias** フィールドがH用されます。

**lu_alias**

ローカル LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を } つので、8 バイトすべてを設定する , 要があります。 **lu_alias** と **lu_name** の両方をすべてゼロに設定すると、verb は、CP と関連Uけられている LU（省略 ~ の LU）に転送されます。

**bypass_security**

機密保護がバイパスされるかどうかをX定します（ AP_YES または AP_NO）。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_INVALID_LU_ALIAS
AP_INVALID_BYPASS_SECURITY

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# CREATE_PASSWORD_SUBSTITUTE

CREATE_PASSWORD_SUBSTITUTE はX定されたセッションに置換文 z と検査装置を生成するためにH用されるパスワード置換文 z 、パスワード検査装置、および送信シーケンス番号を戻します。

## VCB 構造体

```
typedef struct create_password_substitute
{
   unsigned short  opcode;            /* verb operation code        */
   unsigned char   reserv2;           /* reserved                   */
   unsigned char   format;            /* format                     */
   unsigned short  primary_rc;        /* primary return code        */
   unsigned long   secondary_rc;      /* secondary return code      */
   unsigned char   lu_alias[8];       /* LU alias                   */
   unsigned char   conv_group_id[8];  /* partner LU alias           */
   unsigned char   user_id[10];       /* user ID                    */
   unsigned char   pw[10];            /* clear text password        */
   unsigned char   seq_no[8];         /* sequence number            */
   unsigned char   pw_sub[10];        /* password substitute        */
   unsigned char   pw_verifier[10];   /* password verifier          */
} CREATE_PASSWORD_SUBSTITUTE;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_CREATE_PASSWORD_SUBSTITUTE

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_alias**
ローカルに定義された LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。

**conv_group_id**
LU がH用するセッションの会話グループ識別R。

**user_id**
ユーザー ID。

**pw** 暗号化アルゴリズムでH用されるクリア・テキスト・パスワード。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**seq_no**
暗号化アルゴリズムでH用される送信シーケンス番号。verb が成功した場合には、このセッションの送信シーケンス番号の内 t 値は増分されることに注意してください。戻される値は増分後の値です。

**pw_sub**
暗号化アルゴリズムが生成したパスワード置換文 z 。

**pw_verifier**
暗号化アルゴリズムが生成したパスワード検査装置。

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_BAD_LU_ALIAS

AP_DEACT_CG_INVALID_CGID

セッションがパスワード置換をサポートしないために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_PW_SUB_NOT_SUPP_ON_SESS

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_LU_LU_PASSWORD

DEFINE_LU_LU_PASSWORD では、ローカル LU とパートナー LU の間のセッション・レベルの検査にH用するパスワードをX定します。

## VCB 構造体

```
typedef struct define_lu_lu_password
{
   unsigned short  opcode;               /* verb operation code        */
   unsigned char   reserv2;              /* reserved                   */
   unsigned char   format;               /* format                     */
   unsigned short  primary_rc;           /* primary return code        */
   unsigned long   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 /* LULU verification protocol */
   unsigned char   description[RD_LEN];  /* resource description       */
   unsigned char    reserv3[8];          /* reserved                   */
   unsigned char   password[8];          /* password                   */
} DEFINE_LU_LU_PASSWORD;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_LU_LU_PASSWORD

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

ローカル LU の LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文z列です。このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、**lu_alias** フィールドがH用されます。

**lu_alias**

ローカル LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を}つので、8 バイトすべてを設定する，要があります。 **lu_alias** と **lu_name** の両方をすべてゼロに設定すると、verb は、CP と関連Uけられている LU（省略~の LU）に転送されます。

**fqplu_name**

パートナー LU の完全修飾名。この名前の長さは 17 バイトで、右に EBCDIC のスペースが埋められます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文zの文z列で構成されます。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**verification_protocol**

このパートナー LU でH用する LU-LU 検査プロトコル。

**AP_BASIC_PROTOCOL**
このパートナー LU では基本プロトコルだけがH用されます。

**AP_ENHANCED_PROTOCOL**
このパートナー LU ではH張プロトコルだけがH用されます。

**AP_EITHER_PROTOCOL**
このパートナー LU では、以下の基準に基づいて、基本プロトコルとH張プロトコルのいずれをH用することも可能です。

- このフィールドの省略～設定は AP_EITHER_PROTOCOL です。
- AP_EITHER_PROTOCOL の値を設定すると、H張プロトコルへの移行をスムーズに行えます。ローカル LU は、パートナー LU がH張プロトコルを実行することで合意するまで基本プロトコルを受け入れます。いったんH張プロトコルに移行したら、その後 DEFINE_LU_LU_PASSWORD が発行されない限り、基本プロトコルはH用できなくなります。

**description**
q 源の説明。

**password**
パスワード。これは 8 バイトの 16 進数ストリングです。パスワード内の F バイトの最下位ビットは、セッション・レベルの検査ではH用されません。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PLU_NAME

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# DEFINE_USERID_PASSWORD

DEFINE_USERID_PASSWORD では、ユーザー ID と関連Uけられるパスワードを定義します。

## VCB 構造体

```
define_userid_password
{
   unsigned short  opcode;               /* verb operation code        */
   unsigned char   reserv2;              /* reserved                   */
   unsigned char   format;               /* format                     */
   unsigned short  primary_rc;           /* primary return code        */
   unsigned long   secondary_rc;         /* secondary return code      */
   unsigned short  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[RD_LEN];  /* resource description       */
   unsigned short  profile_count;        /* number of profiles         */
   unsigned short  reserv1;              /* reserved                   */
   unsigned char   password[10];         /* password                   */
   unsigned char   profiles[10][10];     /* profiles                   */
} USERID_PASSWORD_CHARS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_USERID_PASSWORD

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**define_type**

定義するユーザー・パスワードのタイプをX定します。

**AP_ADD_USER**

新規ユーザーをX定するか、既存ユーザーのパスワードを変更します。

**AP_ADD_PROFILES**

既存ユーザーのプロファイルに追加します。

**user_id**

ユーザー識別R。これは、10 バイトのタイプ AE の EBCDIC 文z列で、右側の余白に EBCDIC のスペースを埋め込みます。

**password_chars.description**

q源の説明。これは、ローカルに=示可能な文zセットによる 16 バイトのストリングです。16 バイトすべてに意味があります。

**password_chars.profile_count**

プロファイルの数。

**password_chars.password**

ユーザーのパスワード。これは、10 バイトのタイプ AEちづ

# DELETE_LU_LU_PASSWORD

DELETE_LU_LU_PASSWORD では、LU-LU パスワードを削除します。

## VCB 構造体

```
typedef struct delete_lu_lu_password
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   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;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_LU_LU_PASSWORD

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**lu_name**

ローカル LU の LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文z列です。このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、**lu_alias** フィールドがH用されます。

**lu_alias**

ローカル LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を}つので、8 バイトすべてを設定する，要があります。 **lu_alias** と **lu_name** の両方をすべてゼロに設定すると、verb は、CP と関連Uけられている LU（省略~の LU）に転送されます。

**fqplu_name**

パートナー LU の完全修飾名。この名前は、17 バイトの長さで、EBCDIC スペースを右に埋め込みます。EBCDIC ドットによって連結される 2 つのタイプ A の EBCDIC 文zの文z列で構成されます。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**

AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PLU_NAME

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_USERID_PASSWORD

DELETE_USERID_PASSWORD では、ユーザー ID に関連Uけられているパスワードを削除します。

## VCB 構造体

```
typedef struct delete_userid_password
{
   unsigned short  opcode;               /* verb operation code        */
   unsigned char   reserv2;              /* reserved                   */
   unsigned char   format;               /* format                     */
   unsigned short  primary_rc;           /* primary return code        */
   unsigned long   secondary_rc;         /* secondary return code      */
   unsigned short  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[RD_LEN];  /* resource description       */
   unsigned short  profile_count;        /* number of profiles         */
   unsigned short  reserv1;              /* reserved                   */
   unsigned char   password[10];         /* password                   */
   unsigned char   profiles[10][10];     /* profiles                   */
} USERID_PASSWORD_CHARS;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DELETE_USERID_PASSWORD

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**delete_type**

削除のタイプをX定します。

**AP_REMOVE_USER**

ユーザー・パスワードとすべての関連プロファイルを削除します。

**AP_REMOVE_PROFILES**

X定のプロファイルを削除します。

**user_id**

ユーザー識別R。これは、10 バイトのタイプ AE の EBCDIC 文z列で、右側の余白に EBCDIC のスペースを埋め込みます。

**password_chars.description**

このフィールドは、この verb を処理する~は無kされます。

**password_chars.profile_count**

プロファイルの数。

**password_chars.password**

このフィールドは、この verb を処理する~は無kされます。

**password_chars.profiles**
ユーザーに関連Uけられるプロファイル。 F プロファイルは、10 バイトのタイプ AE の EBCDIC 文 z 列で、右側の余白に EBCDIC のスペースを埋め込みます。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_NO_PROFILES

AP_UNKNOWN_USER
AP_INVALID_UPDATE_TYPE

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# SIGN_OFF

SIGN_OFF は LU にサインオン・リストから項目を削除するようにX令します。現在は、サインオン・リストからの項目だけが削除されます。verb は、すべての項目が削除されるか、またはU加 sign_off_data 構造体にある項目だけが削除されるかをX定できます。

## VCB 構造体

```
typedef struct query_sign_off
{
   unsigned short  opcode;          /* verb operation code          */
   unsigned char   reserv2;         /* reserved                     */
   unsigned char   format;          /* format                       */
   unsigned short  primary_rc;      /* primary return code          */
   unsigned long   secondary_rc;    /* secondary return code        */
   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   list;            /* signed on to/from list       */
   unsigned char   all_in_list;     /* sign off all entries in list */
   unsigned char   immediate;       /* remove entries immediately   */
   unsigned char   num_entries;     /* number of entries            */
} QUERY_SIGN_OFF;

typedef struct sign_off_data
{
   unsigned char   user_id[10];      /* user ID                      */
   unsigned char   all_profiles;     /* all profiles for this user   */
   unsigned char   profile[10];      /* specific profile             */
} SIGN_OFF_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_CREATE_PASSWORD_SUBSTITUTE

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

LU 名。この名前は、8 バイトのタイプ A の EBCDIC 文z列です。このフィールドをすべてゼロに設定すると、**lu_alias** フィールドが索引値を判別するためにH用されます。

**lu_alias**

ローカルに定義された LU の別名。これは、ローカル=示可能文zセットの 8 バイトの文z列です。このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を } つので、8 バイトすべてを設定する , 要があります。 **lu_name** および **lu_alias** フィールドが両方ともすべてゼロに設定される場合、制御点 (CP) と関連した LU（省略～ LU）がH用されます。

**plu_alias**
パートナー LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。8 バイトすべてが有効であり、すべて設定する , 要があります。このフィールドをすべてゼロに設定すると、**fqplu_name** フィールドが索引値を判別するためにH用されます。

**fqplu_name**
パートナー LU の 17 バイト完全修飾ネットワーク名。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式でX定します。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**list**
サインオン・リストのタイプ。これは AP_SIGNED_ON_TO_LIST にセットされなければなりません。

**AP_SIGNED_ON_TO_LIST**
ローカル LU からリモート LU にサインオンされたユーザーのリスト。リモート LU は、項目がこのリストから削除されたときにそのことを連絡されないことに注意してください。これは現在サポートされている唯一の値です。

**all_in_list**
AP_YES にセットされている場合には、**list** によりX定されたリスト内のすべてのユーザーはサインオフされます。

**immediate**
AP_YES にセットされている場合には、ユーザーは即 ~ に削除されます。AP_NO にセットされている場合には、ユーザーは、サインオフが正常に完了したことをリモート LU がN認すると、削除されます。このフィールドは、**list** が AP_SIGNED_ON_TO_LIST である場合には予約済みになります。

**num_entries**
実際に戻されるエントリーの数。

**all_in_list** が AP_NO である場合には、ユーザーのリストは SIGN_OFF_DATA 構造体のシリーズとして SIGN_OFF VCB にU加されなければなりません。SIGN_OFF_DATA 構造体のパラメーターは次のとおりです。

**user_id**
ユーザー ID。

**all_profiles**
戻すことができたはずの項目の合計数。これは **num_entries** の数より大きくできます。

**profile**
10 バイトの英数 z EBCDIC ストリング。「プログラム」は現在ブランク・プロファイル (10 EBCDIC スペース) だけをサポートすることに注意してください。このフィールドは **list_options** が AP_FIRST_IN_LIST にセットされていると無 k されます。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_ALIAS

AP_INVALID_LU_NAME
AP_INVALID_PLU_NAME
AP_INVALID_USERID
AP_INVALID_PROFILE
AP_INVALID_LIST
AP_INVALID_LIST_OPTION

「プログラム」が正常にプロセスしない SIGN_OFF_DATA **user_id/profile** のどの組み合わせも VCB にU加されたアプリケーションに戻され、 **num_entries** の戻り値は「プログラム」が戻した SIGN_OFF_DATA 項目 (プロセスされない可能性があります) の数です。

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_ALIAS

AP_INVALID_LU_NAME
AP_INVALID_LU_NAME
AP_INVALID_LIST

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ したために verb が実行されない場合には、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# 第11章 APING と CPI-C の verb

この章では、他のノードに対して 『ping』 を実行するための verb、および CPI-C サイド情報の定義、削除、照会にH用する verb について説明します。

# APING

APING によって、管理アプリケーションは、ネットワーク内のリモート LU に対して 『ping』 を実行できます。 **partner_ver_len** フィールドにゼロよりも大きな値を設定すると、そのX定値の長さの検査データ文z列を VCB の末尾に追加して戻すことが可能になります。

パーソナル・コミュニケーションズまたは Communications Serverの APING は内t『サービス・トランザクション・プログラム』としてインプリメントされます。これにはパーソナル・コミュニケーションズまたは Communications Serverの APPC API (*パーソナル・コミュニケーションズ・クライアント/サーバー・コミュニケーション・プログラミング*で説明) をH用します。

## VCB 構造体

```
typedef struct aping
{
   unsigned short  opcode;          /* verb operation code           */
   unsigned char   reserv2;         /* reserved                      */
   unsigned char   format;          /* format                        */
   unsigned short  primary_rc;      /* primary return code           */
   unsigned long   secondary_rc;    /* secondary return code         */
   unsigned char   lu_name[8];      /* local LU name                 */
   unsigned char   lu_alias[8];     /* local LU alias                */
   unsigned long   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                       */
   unsigned short  dlen;            /* length of data to send        */
   unsigned short  consec;          /* number of consecutive sends   */
   unsigned char   fqplu_name[17];  /* fully qualified partner       */
                                    /* LU name                       */
   unsigned char   echo;            /* data echo flag                */
   unsigned short  iterations;      /* number of iterations          */
   unsigned long   alloc_time;      /* time taken for ALLOCATE       */
   unsigned long   min_time;        /* min send/receive time         */
   unsigned long   avg_time;        /* average send/receive time     */
   unsigned long   max_time;        /* max send/receive time         */
   unsigned short  partner_ver_len; /* size of string to receive     */
} APING;
```

## 指定Ｑｉａー？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。

**opcode**

AP_APING

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**lu_name**

APING verb の送信元であるローカル LU の LU 名。この名前は、8 バイト

のタイプ A の EBCDIC 文 z 列です。 このフィールドをすべてゼロに設定すると、ローカル LU の判別のためには、 **lu_alias** フィールドがH用されます。

**lu_alias**
APING verb の送信元であるローカル LU の別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 このフィールドは、**lu_name** フィールドにすべてゼロを設定した場合にのみ有効です。この場合、8 バイトすべてが意味を } つので、8 バイトすべてを設定する , 要があります。**lu_name** および **lu_alias** の両方に 2 進ゼロを設定すると、省略 ~ の（制御点の）LU がH用されます。

**plu_alias**
ローカル・トランザクション・プログラムがパートナー LU を識別するための別名。これは、ローカル = 示可能文 z セットの 8 バイトの文 z 列です。 8 バイトすべてが有効であり、すべて設定する , 要があります。この名前は、構成 ~ にX定したパートナー LU の名前と一致していなければなりません。このパラメーターに 2 進ゼロを設定すると、代わりに **fqplu_name** パラメーターがH用されます。

**mode_name**
H用するモードの名前。 これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列 (先頭は文 z ）です。EBCDIC スペースが右の余白に埋め込まれます。

**tp_name**
実行するトランザクション・プログラムの名前。これは、64 バイトの文 z 列です。 ノード・オペレーター機能は、この文 z 列の文 z セットを検査しません。**tp_name** の値は、リモート LU に構成した値と一致している , 要があります。この文 z 列には通常、EBCDIC の 『APINGD』 を設定します。右側の余白には EBCDIC のスペースを埋め込みます。

**security**
実行するトランザクション・プログラムへのアクセスの妥当性検査を行うために、パートナー LU が , 要とする情報をX定します。

AP_NONE
AP_PGM
AP_SAME
AP_PGM_STRONG

**pwd** **user_id** と関連Uけられるパスワード。これは、10 バイトのタイプ AE の EBCDIC 文 z 列で、右側の余白に EBCDIC のスペースを埋め込みます。 **security** に AP_PGM または AP_PGM_STRONG を設定した場合にのみ , 要です。

**user_id**
パートナー・トランザクション・プログラムにアクセスするために , 要なユーザー ID。これは、10 バイトのタイプ AE の EBCDIC 文 z 列で、右側の余白に EBCDIC のスペースを埋め込みます。 **security** に AP_PGM、AP_PGM_STRONG、または AP_SAME を設定した場合にのみ , 要です。

**dlen** APING トランザクション・プログラムから送信されるデータの長さ。APING は、**dlen** でX定した長さのゼロ文 z 列を送信します。

**consec**
F 反 | ~ に実行する連続送信の数。APING は、ここでX定した数の MC_SEND_DATA verb を発行します。F verb のデータは、**dlen** でX定したバイト数になります。**echo** パラメーターに AP_YES を設定すると、APING は最後の MC_SEND_DATA を AP_SEND_DATA_P_TO_R_FLUSH （フラッシュ受信用意）としてマークし、パートナー APINGD トランザクション・プログラムからのデータが入った応答を待ちます（MC_RECEIVE_AND_WAIT を発行)。**echo** パラメーターに AP_NO を設定した場合、 APING はデータをフラッシュしてN認を待ちます（MC_SEND_DATA を AP_SEND_DATA_CONFIRM としてマーク)。いずれの場合も、ここで説明した順序は SNA チェーンに相当します。

**fqplu_name**
パートナー LU の 17 バイト完全修飾ネットワーク名。この名前は、2 つのタイプ A の EBCDIC 文 z 列を EBCDIC ドットで区切り、右側の余白に EBCDIC スペースを埋め込んだ形式でX定します。(それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。）このフィールドは、**plu_alias** フィールドにすべてゼロを設定した場合にのみ有効です。

**echo** APING トランザクション・プログラムが , 要な量のデータを送信した後に、応答を予期するかどうかをX定します。

AP_YES
AP_NO

**iterations**
APING が発行する連続順序列（**consec** パラメーターで定義したもの）の反 | 数。 SNA 用語をHえば、このパラメーターでは、送信されるチェーンの数を定義することになります。

**partner_ver_len**
管理アプリケーションが受信できるパートナー・トランザクション・プログラムの検査データ文 z 列の最大長。

## 戻りＱｉａー？ー

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

**primary_rc**
AP_OK

**sense_data**
verb が正常に戻されると、ここはゼロになります。

**alloc_time**
リモート・トランザクション・プログラムに対する MC_ALLOCATE が処理を終了するまでにかかった~間（ミリC)。

**min_time**
データ送信反 | にかかった最低~間（ミリC)。このパラメーターには、パ

ートナーが応答するためにかかった～間も含まれます。応答するとは、 **echo** パラメーターの設定によって、データを送信するか、N認を発行するかのいずれかになります。

**avg_time**
データ送信反 | にかかった平均～間（ミリC）。このパラメーターには、パートナーが応答するためにかかった～間も含まれます。応答するとは、 **echo** パラメーターの設定によって、データを送信するか、N認を発行するかのいずれかになります。

**max_time**
データ送信反 | にかかった最高～間（ミリC）。このパラメーターには、パートナーが応答するためにかかった～間も含まれます。応答するとは、 **echo** パラメーターの設定によって、データを送信するか、N認を発行するかのいずれかになります。

**partner_ver_len**
パートナー・トランザクション・プログラムによって戻された検査文 z 列の長さ。検査文 z 列そのものは、VCB の末尾に追加されます。

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_LU_NAME

AP_INVALID_LU_ALIAS

APING は、パーソナル・コミュニケーションズまたは Communications Serverの APPC API に用意されている MC_ALLOCATE、MC_SEND_DATA、MC_RECEIVE_AND_WAIT、MC_CONFIRM、MC_DEALLOCATE の verb をH用します。実行が失敗した～にこれらの verb によって戻されるパラメーターについては *パーソナル・コミュニケーションズ・クライアント/サーバー・コミュニケーション・プログラミング* を 2 照してください。

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# CPI-C の verb

# DEFINE_CPIC_SIDE_INFO

この verb では、メモリー内のサイド情報項目を追加または置換します。CPI-C サイド情報項目は、会話特性セットに記号宛先名を関連Uけるものです。この verb でX定した記号宛先名と同じ記号宛先名のサイド情報項目がメモリー内にすでにある場合、元のデータはこの呼び出しにX定したデータで上書きされます。 パーソナル・コミュニケーションズまたは Communications Serverに用意されている CPI-C サポートの詳細については、「*CPI-C Reference*」を 2 照してください。

## VCB 構造体

```
typedef struct define_cpic_side_info
{
   unsigned short  opcode;               /* verb operation code        */
   unsigned char   reserv2;              /* reserved                   */
   unsigned char   format;               /* format                     */
   unsigned short  primary_rc;           /* primary return code        */
   unsigned long   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;     /* defined data               */
} DEFINE_CPIC_SIDE_INFO;

typedef struct cpic_side_info_def_data
{
   unsigned char   description[RD_LEN];  /* resource description       */
   CPIC_SIDE_INFO  side_info;            /* CPIC side info             */
   unsigned char   user_data[32];        /* User defined data          */
} 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                   */
   unsigned long   tp_name_type;         /* TP name type               */
   unsigned char   tp_name[64];          /* TP name                    */
   unsigned char   mode_name[8];         /* Mode name                  */
   unsigned long   conversation_security_type;
                                         /* Conversation security type */
   unsigned char   security_user_id[CPIC_SECURITY_INFO_LEN];
                                         /* User ID                    */
   unsigned char   security_password[CPIC_SECURITY_INFO_LEN];
                                         /* Password                   */
} CPIC_SIDE_INFO;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DEFINE_CPIC_SIDE_INFO

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**sym_dest_name**

サイド情報項目を識別するための記号宛先名。これはローカル=示可能文zセットの最長 8 バイトの名前で、余白にはスペースを埋め込みます。H用できる文zは、英大文zの A から Z と数zの 0 から 9 です。

**def_data.description**
リソースの記述（QUERY_CPIC_SIDE_INFO で戻されたもの）。これは、ローカルに = 示可能な文 z セットによる 16 バイトのストリングです。 16 バイトすべてに意味があります。

**def_data.side_info.partner_lu_name**
パートナー LU の完全修飾名。これはローカル = 示可能文 z セットの 17 バイトの名前で、右側の余白にはスペースを埋め込みます。2 つの文 z 列をドットで区切った形になります。 (それぞれの名前は、埋め込みスペースがなく、最大 8 バイトの長さです。)

**def_data.side_info.tp_name_type**
トランザクション・プログラム名のタイプ。このフィールドには、以下のいずれかの値が設定されます。

**XC_APPLICATION_TP**
X 定したトランザクション・プログラム名は、サービス・トランザクション・プログラムではありません。トランザクション・プログラム名に H 用するすべての文 z は、ローカル = 示可能文 z セットの有効文 z でなければなりません。

**XC_SNA_SERVICE_TP**
X 定したトランザクション・プログラム名は、サービス・トランザクション・プログラムの名前です。トランザクション・プログラム名に H 用するすべての文 z （先頭文 z を除く）は、ローカル = 示可能文 z セットの有効文 z でなければなりません。先頭文 z は、X'01' から X'3F' の範囲の 16 進数文 z にする , 要があります。ただし、X'0E' と X'0F' は H 用できません。

**def_data.side_info.tp_name**
トランザクション・プログラム名。これは、ローカル = 示可能文 z セットの 64 バイト文 z 列で、右側の余白にはスペースを埋め込みます。

**def_data.side_info.mode_name**
モード名。これは、ローカル = 示可能文 z セットの 8 バイト文 z 列で、右側の余白にはスペースを埋め込みます。

**def_data.side_info.conversation_security_type**
会話機密保護のタイプ。このフィールドには、以下のいずれかの値が設定されます。

XC_SECURITY_NONE
XC_SECURITY_SAME
XC_SECURITY_PROGRAM
XC_SECURITY_PROGRAM_STRONG.

**def_data.side_info.security_user_id**
ユーザー ID。 パーソナル・コミュニケーションズまたは Communications Serverは、このフィールドを H 用して、会話レベルの機密保護を実 \ します。

**def_data.side_info.security_password**
パスワード。パーソナル・コミュニケーションズまたは Communications Server は、このフィールドを H 用して、会話レベルの機密保護を実 \ します。

**def_data.user_data**
ユーザー・データ。このデータは、QUERY_CPIC_SIDE_INFO によって戻されますが、パーソナル・コミュニケーションズまたは Communications Server はこのデータをH用したり解釈したりしません。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_SYM_DEST_NAME

AP_INVALID_LENGTH

ノードがまだ+ Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# DELETE_CPIC_SIDE_INFO

この verb では、CPI-C サイド情報項目を削除します。 パーソナル・コミュニケーションズまたは Communications Serverに用意されている CPI-C サポートの詳細については、「*CPI-C Reference*」を 2 照してください。

## VCB 構造体

```
typedef struct delete_cpic_side_info
{
   unsigned short  opcode;           /* verb operation code        */
   unsigned char   reserv2;          /* reserved                   */
   unsigned char   format;           /* format                     */
   unsigned short  primary_rc;       /* primary return code        */
   unsigned long   secondary_rc;     /* secondary return code      */
   unsigned char   reserv2a[8];      /* reserved                   */
   unsigned char   sym_dest_name[8]; /* Symbolic destination name  */
} DELETE_CPIC_SIDE_INFO;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**
AP_DELETE_CPIC_SIDE_INFO

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**sym_dest_name**
サイド情報項目を識別するための記号宛先名。これはローカル＝示可能文ｚセットの最長 8 バイトの名前で、余白にはスペースを埋め込みます。H用できる文ｚは、英大文ｚの A から Z と数ｚの 0 から 9 です。

## 戻りＱｉａー？ー

verb が正常に実行された場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_OK

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_SYM_DEST_NAME

ノードがまだ＋Oされていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_CPIC_SIDE_INFO

この verb は、X定の記号宛先名のサイド情報項目を戻します。 情報はリストの形式で戻されます。 特定のサイド情報項目または特定の項目群を = 示するには、**sym_dest_name** フィールドを設定する , 要があります。 そうでない場合は、このフィールドにすべてゼロを設定します。

## VCB 構造体

```
typedef struct query_cpic_side_info
{
   unsigned short  opcode;            /* verb operation code        */
   unsigned char   reserv2;           /* reserved                   */
   unsigned char   format;            /* format                     */
   unsigned short  primary_rc;        /* primary return code        */
   unsigned long   secondary_rc;      /* secondary return code      */
   unsigned char   *buf_ptr;          /* pointer to buffer          */
   unsigned long   buf_size;          /* buffer size                */
   unsigned long   total_buf_size;    /* total buffer size required */
   unsigned short  num_entries;       /* number of entries          */
   unsigned short  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
{
   unsigned short  overlay_size;      /* size of this 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
{
   unsigned char   partner_lu_name[17];
                                      /* Fully qualified partner    */
                                      /* LU name                    */
   unsigned char   reserved[3];       /* Reserved                   */
   unsigned long   tp_name_type;      /* TP name type               */
   unsigned char   tp_name[64];       /* TP name                    */
   unsigned char   mode_name[8];      /* Mode name                  */
   unsigned long   conversation_security_type;
                                      /* Conversation security type */
   unsigned char   security_user_id[CPIC_SECURITY_INFO_LEN];
                                      /* User ID                    */
   unsigned char   security_password[CPIC_SECURITY_INFO_LEN];
                                      /* Password                   */
} CPIC_SIDE_INFO;

typedef struct cpic_side_info_def_data
{
   unsigned char   description[RD_LEN];  /* resource description       */
   CPIC_SIDE_INFO  side_info;            /* CPIC side info             */
   unsigned char   user_data[32];        /* User defined data          */
} CPIC_SIDE_INFO_DEF_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_QUERY_CPIC_SIDE_INFO

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**buf_ptr**
リスト情報が書き込まれるバッファーへのポインター。

**buf_size**
提供されるバッファーのサイズ。 戻りデータはこのサイズを超えることはありません。

**num_entries**
戻される項目の最大数。項目の数はこの値を超えません。ゼロという値は限度がないことを意味します。

**list_options**
ここでは、リスト情報として何を戻すかをX定します。 **sym_dest_name** のX定値（下記2照）は、実際に戻されるデータの+O点を示すための索引値になります。

**AP_FIRST_IN_LIST**
索引値は無kされ、戻りリストはリスト内最初の項目から+Oされます。

**AP_LIST_FROM_NEXT**
戻りリストは、システムに提供された索引値が示す項目の次の項目から+Oされます。

**AP_LIST_INCLUSIVE**
戻りリストは、索引値によってX定される項目からOまります。

**sym_dest_name**
サイド情報項目を識別するための記号宛先名。これはローカル=示可能文zセットの最長 8 バイトの名前で、余白にはスペースを埋め込みます。H用できる文zは、英大文zの A から Z と数zの 0 から 9 です。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」は以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**buf_size**
バッファーに戻った情報の長さ。

**total_buf_size**
要求されたすべてのリスト情報を戻すために，要になるバッファー・サイズを示す戻り値。この値は、**buf_size** の値よりも大きいことがあります。

**num_entries**
実際に戻されるエントリーの数。

**total_num_entries**
戻すことができたはずの項目の合計数。この値は、**num_entries** の値よりも大きいことがあります。

**cpic_side_info_data.overlay_size**
この項目内のバイト数。つまり、次の項目までのオフセット（もしあれば)。

**cpic_side_info_data.sym_dest_name**
戻されたサイド情報項目の記号宛先名。

**cpic_side_info_data.def_data**
DEFINE_CPIC_SIDE_INFO verb でX定した定義済みの CPI-C サイド情報。

**注:** パーソナル・コミュニケーションズまたは Communications Serverが DEFINE_CPIC_SIDE_INFO を処理した後は、 CPIC 呼び出しによって、この verb で戻されるサイド情報が変更されていることもあります。

状態エラーのために verb が実行されない場合、「プログラム」は、以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_INVALID_SYM_DEST_NAME

ノードがまだ + O されていないために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

ノードが停 _ しているために verb が実行されなかった場合、「プログラム」は以下のパラメーターを戻します。

**primary_rc**
AP_NODE_STOPPING

システム・エラーのために verb が実行されない場合には、「プログラム」は次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# 第12章 接続^Ｍ－８ｃ－ verb

パーソナル・コミュニケーションズまたは Communications Serverの接続マネージャーは、APPC プログラムまたは CPI-C プログラムの起動を管理するためにH用します。接続マネージャー機能の説明についてはパーソナル・コミュニケーションズ・クライアント/サーバー・コミュニケーションズ・プログラミング

パーソナル・コミュニケーションズまたは Communications Serverのノード・オペレーター機能は、接続マネージャーを制御するための 3 つの verb をサポートしています。これらの verb は、パーソナル・コミュニケーションズまたは Communications Serverのノード・オペレーター機能をH用するアプリケーション・プログラムでもH用可能です。

# DISABLE_ATTACH_MANAGER

パーソナル・コミュニケーションズまたは Communications Serverの接続マネージャーは、省略～設定では、ノードのO動～にH用可能になります。ユーザーはこの verb をH用して、すべての動的ロードをH用T可にできます。この verb をH用すると、接続マネージャーがトランザクション・プログラムの起動前に検査するグローバル・フラグがリセットされます。

## VCB 構造体

```
typedef struct disable_am
{
   unsigned short  opcode;            /* Verb operation code        */
   unsigned char   reserv2;           /* reserved                   */
   unsigned char   format;            /* format                     */
   unsigned short  primary_rc;        /* Primary return code        */
   unsigned long   secondary_rc;      /* Secondary return code      */
} DISABLE_AM;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_DISABLE_ATTACH_MGR

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

verb が正常に実行された場合、接続マネージャーは以下のパラメーターを戻します。

**primary_rc**

AP_OK

ノードがまだ+ Oされていないために verb が実行されなかった場合、接続マネージャーは以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT STARTED

システム・エラーのために verb が実行されなかった場合、接続マネージャーは以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# ENABLE_ATTACH_MANAGER

接続マネージャーがH用T可になっているとき、パーソナル・コミュニケーションズまたは Communications Serverのノード・オペレーター機能の verb である ENABLE_AM を発行すれば再びH用可能にできます。この verb をH用すると、接続マネージャーがトランザクション・プログラムの起動前に検査するグローバル・フラグが設定されます。

## VCB 構造体

```
typedef struct enable_am
{
   unsigned short  opcode;          /* Verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* Primary return code        */
   unsigned long   secondary_rc;    /* Secondary return code      */
} ENABLE_AM
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

AP_ENABLE_ATTACH_MGR

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

verb が正常に実行された場合、接続マネージャーは以下のパラメーターを戻します。

**primary_rc**

AP_OK

ノードがまだ+ Oされていないために verb が実行されなかった場合、接続マネージャーは以下のパラメーターを戻します。

**primary_rc**

AP_NODE_NOT STARTED

システム・エラーのために verb が実行されなかった場合、接続マネージャーは以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# QUERY_ATTACH_MANAGER

QUERY_ATTACH_MANAGER verb をH用して、接続マネージャーの状況を調べることができます。接続マネージャーの + O と停 _ には、 ENABLE_ATTACH_MANAGER コマンドと DISABLE_ATTACH_MANAGER コマンドをH用します。

## VCB 構造体

```
typedef struct query_am
{
   unsigned short  opcode;          /* Verb operation code          */
   unsigned char   reserv2;         /* reserved                     */
   unsigned char   format;          /* format                       */
   unsigned short  primary_rc;      /* primary return code          */
   unsigned long   secondary_rc;    /* secondary return code        */
   unsigned short   active;         /* status of the Attach Manager */
} QUERY_AM;
```

## 指定Ｑｉａー？ー

**opcode**
AP_QUERY_ATTACH_MGR

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

## 戻りＱｉａー？ー

verb が正常に実行された場合、以下のパラメーターが戻されます。

**primary_rc**
AP_OK

**active** このフィールドは、接続マネージャーの状況を報告します。

**AP_YES**
接続マネージャーはアクティブです。

**AP_NO**
接続マネージャーはアクティブではありません。

パラメーター・エラーのために verb が実行されなかった場合、以下のパラメーターが戻されます。

**primary_rc**
AP_PARAMETER_CHECK

ノードがまだ + O されていないために verb が実行されなかった場合、接続マネージャーは以下のパラメーターを戻します。

**primary_rc**
AP_NODE_NOT_STARTED

システム・エラーのために verb が実行されなかった場合、接続マネージャーは以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# 第2部 Ｑ一＝Ｊｋ・３＿ｅＫ１一７ｇｓ：および Communications Server管理５一Ｓ９ API

# 第13章 管理５ーＳ９ API の紹介

この t では、パーソナル・コミュニケーションズまたは Communications Server に用意されている管理サービス API について説明します。

## 管理５ーＳ９ verb

パーソナル・コミュニケーションズまたは Communications Server は、以下の管理サービス (MS) verb をサポートしています。これによって、アプリケーション・プログラムは、 SNA ネットワーク内にある管理サービスのフォーカル・ポイントに潜在的な問題を報告できます。

- ALERT_INDICATION
- FP_INDICATION
- MDS_MU_RECEIVED
- NMVT_RECEIVED
- SEND_MDS_MU
- TRANSFER_MS_DATA

## ( ｓＨｊー・ ] $ ｓＨ

パーソナル・コミュニケーションズまたは Communications Server には、管理サービスの verb を処理するためのライブラリー・ファイルが用意されています。

管理サービスの verb には、わかりやすい言語インターフェースがあります。 *verb 制御ブロック*と呼ばれるメモリー・ブロック内のフィールドには、プログラマー作成のプログラムによって情報が書き込まれます。そのプログラムはエントリー・ポイントを呼び出して、 verb 制御ブロックへのポインターを渡します。この操作が終わると、管理サービス (MS) の API は、H用した修正済みのフィールドを verb 制御ブロックに戻します。プログラムは、verb 制御ブロックからの戻りパラメーターを読み取ることができます。管理サービスの verb のエントリー・ポイントは以下のとおりです。

- ACSSVC()
- WinMS()
- WinAsyncMS()
- WinAsyncMSEx()
- WinCSV()
- WinMSCancelAsyncRequest()
- WinMSCleanup()
- WinMSStartup()

エントリー・ポイントの詳細な説明については、 661ページの『第14章 管理サービスのエントリー・ポイント』を 2 照してください。

## verb 制御ＶｍＣ／ (VCB)

***プログラミングについての注:*** 基本オペレーティング・システムは、呼び出し側アプリケーションのアドレス空間でいくつかのサブシステムを実行することによって、パフォーマンスを最適化します。そのため、十分にあるいは正しくデバッグされていないアプリケーション・プログラムによってローカル記述Rテーブル (LDT) の セレクターが誤用されると、操作に誤りが生じたり、システム障 2 が発生したりするおそれがあります。 したがって、アプリケーション・プログラムでは、ポインターの LDT セレクター・フィールドが変更されてしまうようなポインター；術演；を実行するべきではありません。

verb 制御ブロック (VCB) のためにH用するセグメントは、読み書き可能なデータ・セグメントである，要があります。プログラマー作成のプログラムでは、VCB をプログラム内の変数として宣言できます。あるいは、プログラム内に VCB をd り当てたり、もっと大きなセグメントから VCB をd り当てたりすることも可能です。 VCB は、プログラムが発行する verb のためのすべてのフィールドを収容できるだけの大きさにする，要があります。

verb が発行されてから処理が終了するまで、アプリケーション・プログラムでは、verb 制御ブロックの設定を変更するべきではありません。管理サービスは verb の処理を終了すると、変更済みの完成した VCB を元のブロックにコピーして戻します。したがって、読者のプログラムで verb 制御ブロックを変数として宣言する場合は、内 t プロシージャーのスタック内よりも静的記憶装置内に宣言するほうがよいでしょう。

F VCB 内の予約済みの未H用フィールドにはすべてゼロ (X'00') を入れてください。実際のところ、verb 制御ブロック全体にゼロをX定してから、プログラムでパラメーターに値をd り当てるほうが~間の節約になる場合があります。特に、予約済みフィールドにゼロを設定することは重要です。

**注:** VCB が読み書き可能ではない場合、または少なくとも 10 バイトの大きさ（管理サービスの 1 次戻りコードと 2 次戻りコードを収容できる大きさ）がない場合、管理サービスは VCB にアクセスできないため、ベース・オペレーティング・システムの処理が異常終了します。この終了は、*一般保護違反*、プロセッサー例 0 トラップ D として認識されます。

VCB が小さすぎる場合、または間違った種類のセグメントがH用されている場合は、管理サービスが INVALID_VERB_SEGMENT という 1 次戻りコードを戻します。

## 管理５ーＳ９ (MS) Ｗｍ０ｉ｀の作成

パーソナル・コミュニケーションズまたは Communications Serverには、管理サービスの verb を処理するためのダイナミック・リンク・ライブラリー (DLL) ファイルが用意されています。

DLL は再入可能です。つまり、複数のアプリケーションのプロセスとスレッドが同~に DLL を呼び出せます。

管理サービスの verb には、わかりやすい言語インターフェースがあります。 *verb 制御ブロック* (VCB) と呼ばれるメモリー・ブロック内のフィールドには、プログラマー作成のプログラムによって情報が書き込まれます。そのプログラムは管理サービスの DLL を呼び出して、 verb 制御ブロックへのポインターを渡します。この操作が終わると、管理サービスは、H用した更新済みのフィールドを VCB に戻します。プログラムは、verb 制御ブロックからの戻りパラメーターを読み取ることができます。

659ページの = 3 では、管理サービス・プログラムのコンパイルとリンクに，要なシステム提供のヘッダー・ファイルとライブラリーのH用方法をソース・モジュールごとにまとめています。一 t のヘッダー・ファイルには、他の，要なヘッダー・ファイルが入っている場合があります。

*= 3. 管理サービスのヘッダー・ファイルとライブラリー*

| オペレーティング・システム | ヘッダー・ファイル | ライブラリー | DLL 名 |
|---|---|---|---|
| WINNT & WIN95 | WINMS.H | WINMS32.LIB | WINMS32.DLL |
| WIN3.1 | WINCSV.H | WINCSV.LIB | WINCSV.DLL |
| OS/2 | ACSSVCC.H | ACSSVC.LIB | ACSSVC.DLL |

## SNA API / i $ " s H ・ 5 ] ー H

Windows 95、Windows NT、Windows 3.1、および OS/2 オペレーティング・システム用のクライアントのセットが Communications Server に組み込まれています。これらのクライアントのことを本書では「SNA API クライアント」と呼びます。SNA API クライアントは、管理サービス verb のサブセットだけをサポートしています。具体的には次のとおりです。

- **WINMS** は、Windows 95 および NT クライアントでサポートされる唯一の API です。詳細は664ページの『WinMS()』を 2 照してください。
- **WINCSV** は Windows 3.1 クライアントでのみサポートされます。詳細は663ページの『WinCSV()』を 2 照してください。
- **ACSSVC** は SNA API OS/2 クライアントでのみサポートされます。詳細は662ページの『ACSSVC()』を 2 照してください。

サポートされている管理サービスの verb のリストを以下に示します。

- TRANSFER_MS_DATA
- SEND_MDS_MU

# 第14章 管理５ーＳ９の（ｓＨｊー・］＄ｓＨ

この章では、管理サービス verb のエントリー・ポイントについて説明します。

# ACSSVC()

これは SNA API OS/2 クライアント用にサポートされる唯一のエントリー・ポイントです。

ここには、OS/2 SNA API クライアントで管理サービス API の以下の verb を発行するための同期エントリー・ポイントが用意されています。

## 構文

```
void ACSSVC (long);
```

入力は verb 制御ブロック・ポインターです。

## 戻り値

戻り値の 1 次と 2 次の戻りコードを検査してください。

# WinCSV()

これは Windows 3.1 クライアント用にサポートされる唯一のエントリー・ポイントです。

このファンクションには CSV API 用の同期エントリー・ポイントが用意されています。

## 構文

```
void WINAPI WinCSV(long vcb)
```

| パラメーター | 説明 |
|---|---|
| **vcb** | verb 制御ブロックへのポインター。 |

## 戻り値

戻り値はありません。verb 制御ブロックの **primary_rc** および **secondary_rc** フィールドに値が入っている場合は、エラーが発生しています。

# WinMS()

これは Windows 95 および Windows NT 用にサポートされる唯一のエントリー・ポイントです。

ここには、管理サービスの API の以下の verb を発行するための同期エントリー・ポイントが用意されています。

- SEND_MDS_MU
- TRANSFER_MS_DATA

## 構文

```
void WINAPI WinMS(long vcb, unsigned short vcb_size);
```

| パラメーター | 説明 |
|---|---|
| **vcb** | verb 制御ブロックへのポインター |
| **vcb_size** | verb 制御ブロックのバイト数 |

## 戻り値

戻り値はありません。 verb 制御ブロックの **primary_rc** および **secondary_rc** フィールドに値が入っている場合は、エラーが発生しています。

## 解説

これは、管理サービスの API の主要な同期エントリー・ポイントです。この呼び出しは verb の処理が終了するまでブロックされます。

## WinMSCleanup()

この関数は[redacted]ービスの API から管理サービス・アプ
登録抹消し[redacted]

### 構文

```
BOOL WINAPI [redacted]anup(void);
```

### 戻り値

戻り値によ[redacted]録抹消が正常に実行されたかどうかを
でなければ[redacted]ケーションの登録抹消が正常に実行さ
戻された場[redacted]プリケーションの登録は抹消されてい[redacted]

### 解説

WinMSCl[redacted] 0.361 Tf 0.998 0 T=0+

# WinMSStartup()

この関数によって、アプリケーションは , 要な管理サービス API のバージョンをX定し、製 J がサポートしている API のバージョンを検索できます。この関数は、アプリケーションが登録のための管理サービス API を発行する前に呼び出すことができます。

## 構文

```
int WINAPI WinMSStartup(WORD wVersionRequired,
                        LPWMSDATA msdata);
```

**パラメーター　説明**

**wVersionRequired**
: , 要な管理サービス API サポートのバージョンをX定します。高位バイトでマイナー・バージョン（改訂）をX定し、低位バイトでメジャー・バージョン番号をX定します。

**msdata**
: 管理サービス API サポートのバージョンと、管理サービス・システムの説明を戻します。

## 戻り値

戻り値によって、アプリケーションが正常に登録されたかどうか、および管理サービス API システムがX定のバージョン番号をサポートしているかどうかを判別できます。値がゼロの場合は、アプリケーションが正常に登録されました。また、X定のバージョンはサポートされています。それ以 0 の値は以下のいずれかになります。

**WMSSYSERROR**
: 基礎となるネットワーク・サブシステムが、ネットワーク通信を行える状態になっていません。

**WMSVERNOTSUPPORTED**
: 要求されたバージョンの管理サービスの API サポートが、この管理サービスの API システムでは提供されていません。

**WMSBADPOINTER**
: msdata パラメーターに誤りがあります。

## 解説

WinMSStartup は、API の今後のバージョンとの互換性をN保するためのものです。サポートされている現行バージョンは 1.0 です。

**WinMSStartup** および **WinMSCleanup** のH用は , 須ではありません。しかし、アプリケーションでは、この 2 つの呼び出しのH用が一貫している , 要があります。つまり、両方ともH用するか、両方ともH用しないかのどちらかにしてください。

**注: WinMSCleanup()** も 2 照してください。

# WinMSRegisterApplication()

この関数では、アプリケーションを、NMVT レベル・アプリケーション、MDS レベル・アプリケーション、またはアラート・ハンドラーのいずれかとして登録します。この登録方法によって、アプリケーションが受け取る非送信請求X示の種類が決まります。

- NMVT レベル・アプリケーションは、NMVT_RECEIVED X示を受け取ります。
- MDS レベル・アプリケーションは、MDS_MU_RECEIVED X示を受け取ります。また、フォーカル・ポイントの状況に変化があった場合は、 FP_NOTIFICATION X示を受け取ります。
- アラート・ハンドラーは、ALERT_INDICATION X示を受け取ります。

**注:** NMVT を MDS MU に変換して受け取るように登録することも可能です。

これらのX示を処理できないアプリケーションでは、**WinMSRegisterApplication** を呼び出すべきではありません。

## 構文

```
BOOL WINAPI WinMSRegisterApplication(unsigned short reg_type,
                                     unsigned char  *ms_appl_name,
                                     unsigned short vector_key,
                                     unsigned char  mds_conv_reqd,
                                     unsigned char  *ms_category,
                                     unsigned short max_rcv_size,
                                     unsigned char  alert_dest,
                                     unsigned short *primary_rc,
                                     unsigned long  *secondary_rc);
```

**パラメーター**

**説明**

**reg_type**

登録のタイプ

| | |
|---|---|
| WMSNMVTAPP | NMVT レベル・アプリケーション<br>(または、NMVT を受信するために登録する<br>MDS レベル・アプリケーション ) |
| WMSMDSAPP | MDS レベル・アプリケーション |
| WMSALERTHANDLER | アラート・ハンドラー |

**ms_appl_name**

管理サービスのアプリケーション名。有効な名前は、8 バイト英数 z のタイプ 1134 の EBCDIC 文 z 列（ , 要に応じて後続の余白にスペース (X'40') 文 z が埋め込んだ形式）か、管理サービスの規則固有のアプリケーション・プログラムの名前（後続に余白にスペース (X'40') 文 z が埋め込まれた形式。この名前については、 *SNA Management Services Reference* のU録 D を2照）のいずれかになります。

この名前は、**reg_type** が WMSNMVTAPP または WMSMDSAPP の~にH用されます。 **reg_type** が WMSALERTHANDLER の~は、適用されません。

**vector_key**

アプリケーションにH用できる管理サービスの主要ベクトル・キー。有効な値は以下のとおりです。

| | |
|---|---|
| X'YYYY' | 特定の主要ベクトル・キー |
| AP_SPCF_KEYS | X'8061' から X'8064' までの主要ベクトル・キー |
| AP_ALL_KEYS | すべての主要ベクトル・キー |

このキーは、**reg_type** が WMSNMVTAPP の～にH用されます。**reg_type** が WMSMDSAPP または WMSALERTHANDLER の～は、適用されません。

**mds_conv_reqd**
アプリケーションを MDS レベルとして登録し、受信する NMVT を MDS MU に変換するかどうかをX定します。

（AP_YES または AP_NO）

このパラメーターは、 **reg_type** が WMSNMVTAPP の～にH用されます。**reg_type** が WMSMDSAPP または WMSALERTHANDLER の～は、適用されません。

**ms_category**
アプリケーションで特定の管理サービス・カテゴリーのフォーカル・ポイントに関連した情報を入手したい場合に、そのカテゴリーをX定します。管理サービス・カテゴリーは、管理サービスの規則固有のアプリケーション・プログラムにX定されているカテゴリー・コード（後続の余白にスペース (X'40') 文 z が埋め込まれた形式。 *SNA Management Services Reference* のU録 D を 2 照）か、ユーザー定義のカテゴリーのいずれかになります。ユーザー定義のカテゴリー名は、 8 バイト英数 z のタイプ 1134 の EBCDIC 文 z 列（，要に応じて後続の余白にスペース (X'40') 文 z が埋め込まれた形式）にする，要があります。

このパラメーターは、**reg_type** が WMSMDSAPP の～にH用されます。**reg_type** が WMSNMVTAPP または WMSALERTHANDLER の～は、適用されません。

**max_rcv_size**
アプリケーションがひとまとめに受信できる最大バイト数。このサイズを越えた MDS MU はセグメント化され、F セグメントはそれぞれ別個の MDS_MU_RECEIVED X示によって送信されます。

このパラメーターは、**reg_type** が WMSMDSAPP の～にH用されます。**reg_type** が WMSNMVTAPP または WMSALERTHANDLER の～は、適用されません。

**alert_dest**
このアプリケーションをすべてのアラートの唯一の宛先にするかどうかをX定します。AP_YES と設定すると、すべてのアラートがこのアプリケーションに経路X定され、それ以 0 の所には送信されません。AP_NO と設定すると、アラートはこのアプリケーションに経路X定されるだけではなく、SNA ネットワークを介して通常の方法で送信されます。

このパラメーターは、 **reg_type** が WMSALERTHANDLER の～にH用されます。**reg_type** が WMSNMVTAPP または WMSMDSAPP の～は、適用されません。

**primary_rc**
戻り値: 1 次戻りコード

**secondary_rc**
戻り値: 2 次戻りコード

## 戻り値

この関数の戻り値によって、登録が正常に実行されたかどうかを判別できます。値がゼロでなければ、登録が正常に実行されています。値がゼロの場合は、登録が正常に実行されませんでした。

## 解説

アプリケーションは、複数の呼び出しを行って、複数のクラスのX示を登録できます。

**WinMSRegisterApplication** を呼び出すアプリケーションは、**WinMSGetIndication** を呼び出して、 } ち行列に入っているX示を受け取る , 要があります。

**注: WinMSUnregisterApplication** および **WinMSGetIndication** も 2 照してください。

## WinMSUnregisterApplication()

この関数では、前述の **WinMSRegisterApplication** 呼び出しの効力を取り消して、アプリケーションの登録を

**WinMSUnregisterApplication** と **WinMSCleanup** の違いは以下のとおりです。

- **WinMSUnregisterApplication** では、X示を受け取るための過去の登録を抹消しますが、それ以0の管理サービスの API 呼び出し（WinMS など）を行えないようにするわけではありません。
- **WinMSCleanup** では、管理サービスの API のH用そのものを終了します。

アプリケーションが **WinMSUnregisterApplication** を呼び出した~点で、すでにX示が待ち行列に入っている場合があります。そのようなX示は待ち行列に入ったままになっているので、アプリケーションは、**WinMSGetIndication** を呼び出し、それらのX示を受け取って処理する，要があります。登録が抹消された~以降は、新規のX示が待ち行列に入ることはありません。

**注: WinMSRegisterApplication** および **WinMSGetIndication** も2照してください。

# WinMSGetIndication()

この関数によって、アプリケーションは非送信請求X示を受け取ります。

## 構文

```
int WINAPI WinMSGetIndication(long buffer,
                              unsigned short *buffer_size,
                              unsigned long  timeout);
```

| パラメーター | 説明 |
|---|---|
| **buffer** | X示を受け取るバッファーへのポインター。 |
| **buffer_size** | バッファーのサイズ。戻り値: X示のサイズ。 |
| **timeout** | X示を待機する~間（ミリC）。 |

## 戻り値

この関数の戻り値によって、X示が受け取られたかどうかを判別できます。

**0** X示が戻されました。

**WMSTIMEOUT**
X示の待機~間が切れました。

**WMSSYSNOTREADY**
基礎となるネットワーク・サブシステムが、ネットワーク通信を行える状態になっていません。

**WMSNOTREG**
アプリケーションが、X示を受け取るための登録をしていません。

**WMSBADSIZE**
バッファーが小さすぎるためにX示を受け取れません。十分な大きさがあるバッファーをX定して、 **WinMSGetIndication** 呼び出しを再発行してください。X示のサイズは、**buffer_size** パラメーターで戻されます。

**WMSBADPOINTER**
バッファーまたは **buffer_size** パラメーターのいずれかが無効です。

**WMSSYSERROR**
予期しないシステム・エラーが起きました。

## 解説

これは、ブロック化呼び出しであり、以下のいずれかの状況で値を戻します。

- X示が戻された
- 待機~間が切れた
- アプリケーションが **WinMSCleanup** 呼び出しを発行した
- 製J が停_した
- システム・エラーが起きた

**注: WinMSRegisterApplication** および **WinMSUnregisterApplication** も 2 照してください。

## WinMSGetIndication()

# 第15章 管理５ーＳ９ verb

パーソナル・コミュニケーションズまたは Communications Server で提供される管理サービス API verb をＨえば、アプリケーションはアラートおよび MDS MU を送信したり、ノードが MDS または NMVT データを受信するかアラートを発行するときにＸ示を受け取ることができます。

# TRANSFER_MS_DATA

この verb は、NMVT レベルのアプリケーションが以前に受け取った NMVT 要求に応答するためにH用されます。

MDS レベルのアプリケーションが非送信請求アラートを送信する場合には、TRANSFER_MS_DATA もH用されます。この verb は、WinMS 呼び出しをH用するアプリケーションでH用できます。

## VCB 構造体

```
typedef struct ms_transfer_ms_data
{
   unsigned short  opcode;             /* Verb operation code       */
   unsigned char   data_type;          /* Data type supplied by app */
   unsigned char   format;             /* format                    */
   unsigned short  primary_rc;         /* Primary return code       */
   unsigned long   secondary_rc;       /* Secondary return code     */
   unsigned char   options;            /* Verb options              */
   unsigned char   reserv3;            /* reserved                  */
   unsigned char   originator_id[8];   /* Originator ID             */
   unsigned char   pu_name[8];         /* Physical unit name        */
   unsigned char   reserv4[4];         /* reserved                  */
   unsigned short  dlen;               /* Length of data            */
   unsigned char   *dptr;              /* Data                      */
} MS_TRANSFER_MS_DATA;
```

## 指定Ｑｉａー？ー

アプリケーションは以下のパラメーターをX定します。

**opcode**

SV_TRANSFER_MS_DATA

**data_type**

囲みのあるデータのタイプをX定します。管理サービスは以下の説明にしたがって、データを処理します。H用できる値は、以下のとおりです。

**SV_NMVT**

データには、完全な NMVT 要求単位が含まれます。管理サービスは、データにアラートが含まれ、そのアラートが MDS レベルあるいは移行レベルのフォーカル・ポイントに送信される場合に、データを MDS_MU または CP_MSU に変換します。これは、アプリケーションが NMVT_RECEIVED 信号に応答するときに , 要なタイプです。

**SV_ALERT_SUBVECTORS**

データには、SNA 定義形式による、アラート主要ベクトルの管理サービス・サブベクトルが含まれます。管理サービスでは、NMVT ヘッダーおよびアラート主要ベクトル・ヘッダーが追加されます。その後、管理サービスは、アラートが MDS レベルあるいは移行レベルのフォーカル・ポイントに送信される場合に、データを MDS_MU または CP_MSU に変換します。

**SV_USER_DEFINED**

データには、完全な NMVT 要求単位が含まれます。管理サービスは常にデータをログに記録しますが、送信はしません。

**SV_PDSTATS_SUBVECTORS**
データには、問題判別統計が含まれます。管理サービスは常にデータをログに記録しますが、アラート・ハンドラーが登録済みの場合は、そこに ALERT_INDICATION 中のデータを送ります。

**format**
VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**options**
この verb で提供されるデータに対するオプショナル処理をX定します。管理サービスでは基本的に、X定された **data_type** およびオプション間に対立がなければ、**type** にしたがってデータが処理される点に留意してください。このパラメーターは 1 バイト値であり、個々のビットの設定で選択オプションを示します。すべてのオプションをX定する場合には、このバイトをゼロに設定します。

ビット 0 は最上位ビット、ビット 7 は最下位ビットです。

（**data_type** に SV_USER_DEFINED を設定する場合、ビット 1 から 3 は無 k されます。）

ビット 0: ゼロに設定されると、データに日U／~刻 (X'01') サブベクトルが追加されます。

ビット 1: ゼロに設定すると、データに製 J セット ID (X'10') サブベクトルが追加されます。アプリケーションがすでに製 J セット ID サブベクトルの入ったデータを提供すると、管理サービスは既存サブベクトルの直前に、パーソナル・コミュニケーションズまたは Communications Server の製 J セット ID サブベクトルを追加します。

ビット 2: ゼロに設定すると、SNA セッションにデータが送られます。データにアラートが含まれていない場合、管理サービスは省略~値 SSCP-PU セッションにデータを送ります。データにアラートが含まれていれば、管理サービスは SSCP-PU セッション、CP-CP セッション、または LU-LU セッションのいずれかにデータを送ります。これは、パーソナル・コミュニケーションズまたは Communications Server がアラート・フォーカル・ポイントにアラートを伝送するためにH用するセッションのタイプによって決まります。

ビット 3: ゼロに設定すると、パーソナル・コミュニケーションズまたは Communications Server の問題判別機能によってデータがログに記録されます。

**注:** 管理サービスのヘッダー・ファイルでは以下の定数が提供され、それらの定数は上にX定された個々のビットを 2 照します。

- SV_TIME_STAMP_SUBVECTOR
- SV_PRODUCT_SET_ID_SUBVECTOR
- SV_SEND_ON_SESSION
- SV_LOCAL_LOGGING

ビット 4 ～ 7: 予約済み。

**originator_id**
verb を発行した構成要素の名前。これは、ローカル=示可能文 z セットの 8

バイトの文 z 列です。管理サービスは TRANSFER_MS_DATA をログに記録するときに限って、このフィールドを H 用します。

**pu_name**
データの送信先の PU の名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（右側の余白は、EBCDIC スペースで埋め込みます）に設定します。**pu_name** が X 定されていなければ、すべて 2 進ゼロに設定してください。 TRANSFER_MS_DATA を H って NMVT_RECEIVED 信号に応答するアプリケーションの場合は、NMVT_RECEIVED 信号で受信される **pu_name** を X 定する , 要があります。 **pu_name** を X 定していない場合、タイプ SV_NMVT の TRANSFER_MS_DATA 信号に含まれるデータは、省略 ~ の PU セッションが H 用可能であれば、そのセッションを経由して送られます。アラートを含む TRANSFER_MS_DATA 信号の場合、アプリケーションが特定の PU に対するアラート・データの送信を明示的に X 定しない限り、 **pu_name** は X 定するべきではありません。これを X 定しないことにより、通常の管理サービス・アラート経路 X 定アルゴリズムがバイパスされます。

**dlen** データの長さ。

**dptr** データへのポインター。このポインターを NULL に設定すると、管理サービスは、データが VCB に隣接している（つまり、VCB の直後に続いている）と想定します。

## 戻りＱｉａー？ー

verb が正常に実行された場合、管理サービスは以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されなかった場合、管理サービスは以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
SV_INVALID_DATA_TYPE

SV_DATA_EXCEEDS_RU_SIZE
AP_INVALID_PU_NAME

状態エラーのために verb が実行されなかった場合、管理サービスは以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
SV_SSCP_PU_SESSION_NOT_ACTIVE

システム・エラーのために verb が実行されなかった場合、「プログラム」管理サービスは以下のパラメーターを戻します。

**primary_rc**

AP_UNEXPECTED_SYSTEM_ERROR

# MDS_MU_RECEIVED

管理サービスは以下の場合に、この verb X示を登録済み MDS レベルのアプリケーションに送信します。

- ピア MDS レベルのアプリケーションから MDS_MU を受信した場合。
- NMVT を受信し、かつ以下の場合。
  - 適切な NMVT レベルのアプリケーションが登録されていない。
  - MDS レベル・アプリケーションが、着信 NMVT の管理サービス主要ベクトル・キー内で搬送される名前に対応する名前をHって登録されている（管理サービスは NMVT から MDS_MU への変換を実行します）。

## VCB 構造体

```
typedef struct ms_mds_mu_received
{
   unsigned short  opcode;           /* Verb operation code           */
   unsigned char   reserv2;          /* reserved                      */
   unsigned char   format;           /* format                        */
   unsigned short  primary_rc;       /* Primary return code           */
   unsigned long   secondary_rc;     /* Secondary return code         */
   unsigned char   first_message;    /* First message for curr MDS_MU */
   unsigned char   last_message;     /* Last message for curr MDS_MU  */
   unsigned char   pu_name[8];       /* Physical unit name            */
   unsigned char   reserv3[8];       /* reserved                      */
   unsigned short  mds_mu_length;    /* Length of incoming MDS_MU     */
   unsigned char   *mds_mu;          /* MDS_MU data                   */
} MS_MDS_MU_RECEIVED;
```

## 指定Ｑｉａー？ー

**opcode**

AP_MDS_MU_RECEIVED

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するためには、このフィールドはゼロに設定されます。

**first_message**

これが MDS_MU の最初のメッセージかどうかを示すフラグ（AP_YES または AP_NO）。 **WinMSRegisterApplication** 呼び出しでX定された **max_rcv_size** が送達される MDS_MU の長さよりも短い場合、 MDS_MU はまとめてアプリケーション送信されます。

**last_message**

これが MDS_MU の最後のメッセージかどうかを示すフラグ（AP_YES または AP_NO）。

**pu_name**

NMVT（MDS_MU に変換されている）の発信元である物理装置の名前。着信 NMVT に応答する責任は、アプリケーションにあります。アプリケーションは SEND_MDS_MU をHって応答を送信します。アプリケーションは応答の送信~に、SEND_MDS_MU の **pu_name** フィールドを、

MDS_MU_RECEIVED 信号で提供される **pu_name** に設定する，要があります。 MDS_MU が MDS レベルの転送機構から受信されると、 **pu_name** はすべて 2 進ゼロに設定されます。

**mds_mu_length**
信号に組み込まれた MDS_MU t 分の長さ。

**mds_mu**
MDS_MU データ。データ・ポインターは NULL に設定され、データは VCB に隣接しています（つまり、VCB の直後に続いています）。

# SEND_MDS_MU

この verb は、**WinMS** エントリー・ポイントをHってアラート以0のネットワーク管理データを送信するために MDS レベル・アプリケーションでH用されます。宛先アプリケーションに MDS_MU を送信中にエラーが生じると、そのエラーは以下に示す 2 つのいずれかの方法で起点アプリケーションに戻されます。エラーがローカル・ノードで検出されると、アプリケーションは SEND_MDS_MU 応答の戻りコードによって通知を受けます。エラーがリモート・ノードで検出されると、エラーは MDS_MU_RECEIVED VCB で転送されるエラー MDS_MU によって報告されます。宛先ノードへの送信が SSCP-PU セッションを経由する場合、管理サービスは発信 MDS_MU を NMVT に変換できます。アプリケーションがローカル・ノードの ID を知る，要はありません。 MDS 経路X定情報 GDS 変数の起点位置名サブベクトルの **netid** または **nau**、あるいはその両方のサブフィールドにアプリケーションが 8 個の EBCDIC ブランクをX定した場合、パーソナル・コミュニケーションズまたは Communications Server が適切な値を提供します。アプリケーションが **netid** または **nau** のいずれかに値をX定しないで、かつ 8 個未満のブランクをX定した場合、パーソナル・コミュニケーションズまたは Communications Server は AP_INVALID_MDS_MU_FORMAT の 2 次戻りコードを戻します。

## VCB 構造体

```
typedef struct ms_send_mds_mu
{
   unsigned short  opcode;             /* Verb operation code        */
   unsigned char   reserv2;            /* reserved                   */
   unsigned char   format;             /* format                     */
   unsigned short  primary_rc;         /* Primary return code        */
   unsigned long   secondary_rc;       /* Secondary return code      */
   unsigned char   options;            /* Verb options               */
   unsigned char   reserv3;            /* reserved                   */
   unsigned char   originator_id[8];   /* Originator ID              */
   unsigned char   pu_name[8];         /* Physical unit name         */
   unsigned char   reserv4[4];         /* reserved                   */
   unsigned short  dlen;               /* Length of data             */
   unsigned char   *dptr;              /* Data                       */
} MS_SEND_MDS_MU;
```

## 指定Ｑｉａー？ー

**opcode**

AP_SEND_MDS_MU

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**options**

この verb で提供されるデータに対するオプショナル処理をX定します。このパラメーターは 1 バイト値であり、個々のビットの設定で選択オプションを示します。すべてのオプションをX定する場合には、このバイトをゼロに設定します。

ビット 0 は最上位ビット、ビット 7 は最下位ビットです。

ビット 0: ゼロに設定されると、データに日U／~刻 (X'01') サブベクトルが追加されます。

ビット 1: ゼロに設定すると、データに製 J セット ID (X'10') サブベクトルが追加されます。アプリケーションですでに製 J セット ID サブベクトルの入ったデータが提供されると、管理サービスは既存サブベクトルの直前に、パーソナル・コミュニケーションズまたは Communications Server の製 J セット ID サブベクトルを追加します。

ビット 2: 予約済み。

ビット 3: ゼロに設定すると、パーソナル・コミュニケーションズまたは Communications Server の問題判別機能によってデータがログに記録されます。

**注:** 上に X 定したビット 0、1、3 を 2 照する管理サービスのヘッダー・ファイルでは、以下の定数が提供されます。

- SV_TIME_STAMP_SUBVECTOR
- SV_PRODUCT_SET_ID_SUBVECTOR
- SV_LOCAL_LOGGING

ビット 4: 管理サービスが宛先アプリケーションに管理サービス・データを送信するために省略 ~ の経路 X 定を H 用するか、直接の経路 X 定を H 用するかを X 定します（AP_DEFAULT または AP_DIRECT）。

**注:** ビット 4 を設定する場合は、AP_DEFAULT または AP_DIRECT を正しく桁送りしてください（AP_DIRECT<<3 など）。

ビット 5 ～ 7: 予約済み。

**originator_id**
verb を発行した構成要素の名前。管理サービスは SEND_MDS_MU をログに記録するときに限って、このフィールドを H 用します。

**pu_name**
データの送信先の PU の名前。これは、8 バイト、英数 z 、タイプ A の EBCDIC 文 z 列（右側の余白は、EBCDIC スペースで埋め込みます）に設定します。**pu_name** が X 定されていなければ、すべて 2 進ゼロに設定してください。着信 NMVT から変換された MDS_MU_RECEIVED X 示に応答するために SEND_MDS_MU を H 用するアプリケーションの場合、MDS_MU_RECEIVED 信号で受信される **pu_name** を X 定する , 要があります。 MDS 転送機能を H って転送される MDS_MU では、**pu_name** をすべて 2 進ゼロに設定します。

**dlen** データの長さ。

**dptr** データへのポインター。このポインターを NULL に設定すると、管理サービスは、データが VCB に隣接している（つまり、VCB の直後に続く）と想定します。

## 戻りＱｉａー？ー

verb が正常に実行された場合、「プログラム」管理サービスは以下のパラメーターを戻します。

**primary_rc**
AP_OK

パラメーター・エラーのために verb が実行されなかった場合、「プログラム」管理サービスは以下のパラメーターを戻します。

**primary_rc**
AP_PARAMETER_CHECK

**secondary_rc**
AP_INVALID_PU_NAME

AP_INVALID_MDS_MU_FORMAT
SV_INVALID_DATA_SIZE

状態エラーのために verb が実行されなかった場合、「プログラム」管理サービスは以下のパラメーターを戻します。

**primary_rc**
AP_STATE_CHECK

**secondary_rc**
AP_SSCP_PU_SESSION_NOT_ACTIVE

システム・エラーのために verb が実行されない場合には、「プログラム」管理サービスは次のパラメーターを戻します。

**primary_rc**
AP_UNEXPECTED_SYSTEM_ERROR

# ALERT_INDICATION

この verb X示は、管理サービスが、アラート主要ベクトルを処理する登録済みアラート・ハンドラーまたは登録済み保留アラート・ハンドラーにアラート主要ベクトルを送信するためにH用します。

## VCB 構造体

```
typedef struct ms_alert_indication
{
   unsigned short  opcode;          /* AP_AlERT_INDICATION       */
   unsigned char   reserv2;         /* reserved                  */
   unsigned char   format;          /* format                    */
   unsigned short  primary_rc;      /* Primary return code       */
   unsigned long   secondary_rc;    /* Secondary return code     */
   unsigned short  alert_length;    /* Length of alert           */
   unsigned char   reserv3[6];      /* reserved                  */
   unsigned char   *alert;          /* Alert data                */
} MS_ALERT_INDICATION;
```

## 指定Ｑｉａー？ー

**opcode**

AP_ALERT_INDICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**alert_length**

アラート・データの長さ。

**alert** アラート・データへのポインター。データ・ポインターは NULL に設定され、データは VCB に隣接しています（つまり、VCB の直後に続いています)。

# FP_NOTIFICATION

MDS レベル・アプリケーションが特定の管理サービス・カテゴリーおよびそのカテゴリー変更のフォーカル・ポイントの状況をX定するために登録されていれば、管理サービスはアプリケーションにこの verb 信号を送信します。

## VCB 構造体

```
typedef struct ms_fp_notification
{
   unsigned short  opcode;            /* Verb operation code         */
   unsigned char   reserv2;           /* reserved                    */
   unsigned char   format;            /* format                      */
   unsigned short  primary_rc;        /* Primary return code         */
   unsigned long   secondary_rc;      /* Secondary return code       */
   unsigned char   fp_routing;        /* Type of routing to focal pt */
   unsigned char   reserv1;           /* reserved                    */
   unsigned short  fp_data_length;    /* Length of incoming focal    */
                                      /* point data                  */
   unsigned char    *fp_data;         /* focal point data            */
} MS_FP_NOTIFICATION;
```

## 指定Ｑｉａー？ー

**opcode**

AP_FP_NOTIFICATION

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**fp_routing**

経路X定のタイプ。メッセージをフォーカル・ポイントに送信するときに、SEND_MDS_MU でX定します（AP_DEFAULT または AP_DIRECT）。

**fp_data_length**

フォーカル・ポイント・データの長さ。

**fp_data**

フォーカル・ポイント・データ。フォーカル・ポイント通知 (X'E1') サブベクトルおよびフォーカル・ポイント識別 (X'21') サブベクトルを含みます。このデータ・ポインターは NULL に設定され、データは VCB に隣接しています（つまり、VCB の直後に続いています）。

# NMVT_RECEIVED

管理サービスはリモート・ノードから NMVT を受信したときに、この verb 信号を NMVT レベルの登録済みアプリケーションに送信します。

管理サービスは着信 NMVT の経路X定に際して、以下の規則を適用します。

1. 着信 NMVT で搬送される主要ベクトル・キーをHって登録された NMVT レベル・アプリケーションへの経路X定をnみる。さもなければ...
2. 主要ベクトル・キーが X'8061' から X'8064' までの 1 つであれば、登録済み NMVT レベル AP_SPCF_KEYS アプリケーションへの経路X定をnみる。さもなければ...
3. 登録済み NMVT レベル AP_ALL_KEYS アプリケーションへの経路X定をnみる。さもなければ...
4. NMVT（MDS_MU に変換後）の MDS レベル・アプリケーションへの経路X定をnみる。さもなければ...
5. 主要ベクトル・キーが X'8061' から X'8064' までの 1 つであれば、NMVT（MDS_MU に変換後）の登録済み MDS レベル・アプリケーションへの経路X定をnみる。さもなければ...
6. （MDS_MU に変換後）登録済み AP_ALL_KEYS MDS レベル・アプリケーションへの経路X定をnみる。さもなければ...
7. NMVT に否定応答する。

## VCB 構造体

```
typedef struct ms_nmvt_received
{
   unsigned short  opcode;          /* Verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* Primary return code        */
   unsigned long   secondary_rc;    /* Secondary return code      */
   unsigned char   pu_name[8];      /* Physical unit name         */
   unsigned char   reserv3[6];      /* reserved                   */
   unsigned short  nmvt_length;     /* Length of incoming NMVT    */
   unsigned char   *nmvt;           /* NMVT data                  */
} MS_NMVT_RECEIVED;
```

## 指定Ｑｉａー？ー

**opcode**

AP_NMVT_RECEIVED

**format**

VCB の 形式を識別します。上記にリストされた VCB のバージョンをX定するには、このフィールドをゼロに設定します。

**pu_name**

NMVT の発信元である物理装置の名前。着信 NMVT に応答する責任は、アプリケーションにあります。アプリケーションは TRANSFER_MS_DATA をHって応答を送信します。アプリケーションは応答の送信~に、TRANSFER_MS_DATA の **pu_name** フィールドを、NMVT_RECEIVED 信号で提供される **pu_name** に設定する，要があります。

## NMVT_RECEIVED

**nmvt_length**
NMVT データの長さ。

**nmvt** REGISTER_NMVT_APPLICATION でX定されるタイプの管理サービス主要ベクトルを含んだ完全な NMVT。このデータ・ポインターは NULL に設定され、データは VCB に隣接しています（つまり、VCB の直後に続いています)。

# 付録A. IBM APPN MIB 表

次の = には、RFC1593 で定義されたとおり、IBM APPN 管理情報ブロック (MIB) の = の実行に関する詳細が示されています。この = では、以下のものが定義されています。

- 個々の MIB = を実行するためにH用されるノード・オペレーター機能の照会 verb
- 入力パラメーターの設定値
- ，要とされるフィルター操作

（戻りパラメーターと MIB = 変数の間のマッピングは、ノード・オペレーター機能の照会 verb の定義から取り出すことができます)。パーソナル・コミュニケーションズまたは Communications Server では現在、ibmappnNodePortDlcTraceTable および ibmappnLsStatusTable の MIB = はサポートされていません。

| **IBM MIB 表** | **ノード・オペレーター！能 verb および MIB 表**Q t | **入O パラメーターの設定値** |
|---|---|---|
| ibmappnNodePortTable | QUERY_PORT | **port_name** ibmappnNodePortName |
| ibmappnNodePortIpTable | （注 1） | |
| ibmappnNodePortDlsTable | QUERY_PORT<br>（**dlc_type** で項目を選択） | **port_name**ibmappnNodePortDlsName |
| ibmappnNodePortTrTable | QUERY_PORT | **port_name** ibmappnNodePortTrName |
| ibmappnNodeLsTable | QUERY_LS | **ls_name** ibmappnNodeLsName |
| ibmappnNodeLsIpTable | （注 1） | |
| ibmappnNodeLsDlsTable | QUERY_LS<br>（**dlc_type** で項目を選択） | **ls_name** ibmappnNodeLsDlsName |
| ibmappnNodeLsTrTable | QUERY_LS | **ls_name** ibmappnNodeLsTrName |
| ibmappnNnTopoRouteTable | QUERY_COS | **cos_name** ibmappnNnTopoRouteCos |
| ibmappnNnAdjNodeTable | QUERY_ADJACENT_NN | **adj_nncp_name**<br>ibmappnNnAdjNodeAdjName |
| ibmappnNnTopologyTable | QUERY_NN_TOPOLOGY_NODE | **node_name**<br>ibmappnNnNodeName<br>**node_type**<br>AP_LEARN_NODE<br>**frsn** 0 |

| IBM MIB 表 | ノード・オペレーター！能 verb および MIB 表Q t | 入Oパラメーターの設定値 |
|---|---|---|
| ibmappnNnTgTopologyTable | QUERY_NN_TOPOLOGY_TG | **owner** ibmappnNnTgOwner<br>**owner_type** AP_LEARN_NODE<br>**dest** ibmappnNnTgDest<br>**dest_type** AP_LEARN_NODE<br>**tg_num** ibmappnNnTgNum<br>**frsn** 0 |
| ibmappnNnTopologyFRTable | QUERY_NN_TOPOLOGY_NODE | **node_name** ibmappnNnFRNode<br>**node_type** AP_LEARN_NODE<br>**frsn** ibmappnNnFRFrsn |
| ibmappnNnTgTopologyFRTable | QUERY_NN_TOPOLOGY_TG | **owner** ibmappnNnTgFROwner<br>**owner_type** AP_LEARN_NODE<br>**dest** ibmappnNnTgFRDest<br>**dest_type** AP_LEARN_NODE<br>**tg_num** ibmappnNnTgFRNum<br>**frsn** ibmappnNnTgFRFrsn |
| ibmappnLocalTgTable | QUERY_LOCAL_TOPOLOGY | **dest** ibmappnLocalTGDest<br>**dest_type** AP_LEARN_NODE<br>**tg_num** ibmappnLocalTgNum |
| ibmappnLocalEnTable | QUERY_LOCAL_TOPOLOGY<br>（固有の **dest** をHって項目を選択）<br>（注 2） | **dest** ibmappnLocalEnName<br>**dest_type** AP_END_NODE<br>**dest_type** AP_LEARN_NODE |

| IBM MIB 表 | ノード・オペレーター！能 verb および MIB 表Q t | 入Oパラメーターの設定値 |
|---|---|---|
| ibmappnLocalEnTgTable | QUERY_LOCAL_TOPOLOGY（注 3） | **dest** ibmappnLocalEnTgOrigin<br>**dest_type** AP_LEARN_NODE<br>**tg_num** ibmappnLocalEnTgNum |
| ibmappnDirTable | QUERY_DIRECTORY_LU | **lu_name** ibmappnDirLuName |
| ibmappnCosModeTable | QUERY_MODE_TO_COS_MAPPING | **mode_name** ibmappnCosModeName |
| ibmappnCosNameTable | QUERY_COS | **cos_name** ibmappnCosName |

**注:**

1. パーソナル・コミュニケーションズまたは Communications Server では、IP は DLC タイプとしてサポートされません。
2. 同じ **dest** がある項目は、QUERY_LOCAL_TOPOLOGY で連続して配列されます。
3. ibmappnLocalEnTgTable はU加されたエンド・ノードの k 点から TG を（つまり、エンド・ノードの TG として）=示します。ただし、APPN アーキテクチャーの現行レベルに適合するネットワーク・ノードは、エンド・ノードの TG 情報そのものと直接U加されたエンド・ノードとの間の TG に関する情報のみを保管します。したがって、この = のすべての項目には、ローカル・ノードの名前 (ibmappnNodeCpName) に設定された ibmappnLocalEnTgDest があります。

# 付録B. 特記事項

本書において、日本では発 = されていない IBM 製 J (機 # およびプログラム )、プログラミング、またはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製 J 、プログラミング、またはサービスを日本で発 = する意図があることを , ずしも示すものではありません。本書で、IBM ライセンス・プログラムまたは他の IBM 製 J に言及している t 分があっても、このことは当 : プログラムまたは製 J のみが H 用可能であることを意味するものではありません。これらのプログラムまたは製 J に代えて、IBM の知的所有権を侵 2 することのない機能的に同等な他社のプログラム、製 J またはサービスを H 用することができます。ただし、IBM によって明示的に X 定されたものを除き、これらのプログラムまたは製 J に関する稼動の > 価および検証はお客様の責任で行っていただきます。

IBM および他社は、本書で説明する主題に関する特許権（特許出願を含む）、商 8 権、または著作権を所有している場合があります。本書は、これらの特許権、商 8 権、および著作権について、本書で明示されている場合を除き、実 \ 権、 H 用権等を許諾することを意味するものではありません。実 \ 権、 H 用権等の許諾については、下記の宛先に、書面にてご照会ください。

〒106-0032 東京都港区六本木3丁目2-31
AP v 業所
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

本プログラムのライセンス保 } 者で、(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 , 要とする方は、下記に連絡してください。

IBM Corporation
Department TL3B/062
P.O. Box 12195
Research Triangle Park, NC 27709-2195
U.S.A.

本プログラムに関する上記の情報は、適切な条件の下で H 用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス q 料は、IBM 所定のプログラム契約の契約条項に基づいて IBM より提供されます。

他社の製 J に関する情報は、それらの製 J の提供者、それらの製 J の発 = q 料、またはその他の一般に入手可能な情報源から入手しました。 IBM はそれらの製 J をテストしておらず、パフォーマンスの精度、互換性、またはその他の他社製 J に関するいかなる記述をも保証するものではありません。他社製 J の能力に関するご質問は、それらの製 J の提供者に送るようお願い致します。

著作権 = 示

本書には、様々なオペレーティング・プラットフォームのプログラミング手法を例示するソース言語で書かれたサンプル・アプリケーション・プログラムが掲載されています。このサンプル・プログラムは、サンプル・プログラムが書か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 + 発、 H 用、販売、または配 [ を目的として、いかなる形式においても IBM に対価を Y 払うことなくこれを複製し、改変し、配 [ することができます。これらの例は、すべての場合について完全にテストされたものではありません。 IBM はこれらのプログラムの信頼性、可用性、および機能について法律上の瑕疵担保責任を含むいかなる明示または暗示の保証責任も i いません。

いかなる場合であれ、サンプル・プログラム、またはサンプル・プログラムを改変した二次的著作物をその一 t に含む複製物については、これを第 0 者に配 [ する場合、下記の著作権 = 示を行ってください。

(c) (貴社名) (年) このコードは一 t 分、IBM Corp. のサンプル・プログラムを H 用しています。
(C)Copyright IBM Corp. (年) All rights reserved.

# 付録C. 商標

下記の用語は、米国およびその他の国における IBM 社の商 8 です。

| | |
|---|---|
| ACF/VTAM | MVS |
| Advanced Peer-to-Peer Networking | MVS/ESA |
| AFP | MVS/XA |
| AIX | NetView |
| AnyNet | Operating System/2 |
| APPN | OS/2 |
| AS/400 | OS/400 |
| AT | RACF |
| CICS | System/370 |
| Common User Access | Virtual Machine/Enterprise Systems Architecture |
| IBM | VM/ESA |
| IMS | VTAM |

二重アスタリスク (**) をU けて示されているその他の会社名、製 J 名、およびサービス名は、 F 社の商 8 または登録商 8 です。

C-bus は Corollary, Inc の商 8 です。

ActionMedia、LANDesk、MMX、Pentium および ProShare は米国およびその他の国における Intel Corporation の登録商 8 です。

Java および HotJava は Sun Microsystems, Inc. の商 8 です。

Microsoft、Windows、Windows NT および Windows 95 のロゴは Microsoft Corporation の登録商 8 です。

UNIX は、X/Open カンパニーリミテッドがライセンスしている米国ならびに他の国における登録商 8 です。

PC Direct は Ziff Communications Company の商 8 であり、IBM Corporation がライセンスをH用しています。

# 索引

日本語, 英ｚ, 数ｚ, 特殊文ｚの順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。



## ［ア行］



## ［カ行］



## ［サ行］



## ［ナ行］

## ［ハ行］



## ［ヤ行］



## ［ラ行］



## A



## C



## D



## E



## F



## H



## I



## L



## M



## N

# P



# Q



# R



# S



# T



# V

# W



# X

Printed in Japan

SC88-5631-01