Personal Communications
AS/400 and 3270 for Windows** 95

IBM

System Management Programming

Version 4.1

SC31-8199-00

Personal Communications
AS/400 and 3270 for Windows** 95

IBM

System Management Programming

Version 4.1

SC31-8199-00

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page vii.

First Edition (March 1996)

This edition applies to Version 4.1 Personal Communications and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

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

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department CGM
P.O. Box 12195
Research Triangle Park, North Carolina
27709-2195

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

© **Copyright International Business Machines Corporation 1989, 1996. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices . . . vii
Trademarks . . . viii

About This Book . . . ix
Who Should Read This Book . . . ix
How to Use This Book . . . ix
Double-Byte Character Set Support . . . x
Where to Find More Information . . . xi

Part 1. Personal Communications Node Operator Facility . . . 1

Chapter 1. Introduction . . . 5
Purpose of the Document . . . 5
Personal Communications Node Operator Facility . . . 5
Entry Points . . . 5
Verb Control Blocks (VCB) . . . 6
Compiling and Linking Your Node Operator Facility Program . . . 6
DLL Interface . . . 7
Using Personal Communications High-Level Language Header Files . . . 7

Chapter 2. Overview . . . 9
Overview of the Verbs in This Book . . . 9
How To Read Verb Descriptions . . . 9
Supplied Parameters . . . 9
Returned Parameters . . . 9
Common VCB Fields . . . 10
Configuring the Node for LEN Node support . . . 15
DLC Processes, Ports, and Link Stations . . . 15

Chapter 3. Node Operator Facility Entry Points . . . 17
WinNOF() . . . 18
WinAsyncNOF() . . . 19
WinAsyncNOFEx() . . . 20
WinNOFCancelAsyncRequest() . . . 21
WinNOFCleanup() . . . 22
WinNOFStartup() . . . 23
WinNOFRegisterIndicationSink() . . . 24
WinNOFUnregisterIndicationSink() . . . 25
WinNOFGetIndication() . . . 26

Chapter 4. Node Configuration Verbs . . . 27
DEFINE_ADJACENT_LEN_NODE . . . 28
DEFINE_CN . . . 31
DEFINE_COS . . . 35
DEFINE_DEFAULTS . . . 41
DEFINE_DEFAULT_PU . . . 43
DEFINE_DLC . . . 45
DEFINE_DLUR_DEFAULTS . . . 48
DEFINE_FOCAL_POINT . . . 50

© Copyright IBM Corp. 1989, 1996

DEFINE_INTERNAL_PU 54
DEFINE_LOCAL_LU 57
DEFINE_LS 60
DEFINE_LU_0_TO_3 70
DEFINE_MODE 73
DEFINE_PARTNER_LU 76
DEFINE_PORT 79
DEFINE_TP 85
DELETE_ADJACENT_LEN_NODE 88
DELETE_CN 90
DELETE_COS 92
DELETE_DLC 94
DELETE_FOCAL_POINT 96
DELETE_INTERNAL_PU 98
DELETE_LOCAL_LU 100
DELETE_LS 102
DELETE_LU_0_TO_3 104
DELETE_MODE 106
DELETE_PARTNER_LU 108
DELETE_PORT 110
DELETE_TP 112

Chapter 5. Activation and Deactivation Verbs 115
START_DLC 116
START_INTERNAL_PU 118
START_LS 121
START_PORT 123
STOP_DLC 125
STOP_INTERNAL_PU 127
STOP_LS 129
STOP_PORT 131
ACTIVATE_SESSION 133
DEACTIVATE_CONV_GROUP 136
DEACTIVATE_SESSION 138
PATH_SWITCH 141

Chapter 6. Query Verbs 143
QUERY_CN 144
QUERY_CN_PORT 149
QUERY_COS 152
QUERY_DEFAULT_PU 156
QUERY_DEFAULTS 158
QUERY_DIRECTORY_LU 160
QUERY_DIRECTORY_STATS 164
QUERY_DLC 166
QUERY_DLUR_LU 171
QUERY_DLUR_PU 175
QUERY_DLUS 181
QUERY_FOCAL_POINT 186
QUERY_LOCAL_LU 191
QUERY_LOCAL_TOPOLOGY 196
QUERY_LS 201
QUERY_LU_0_TO_3 214
QUERY_MDS_APPLICATION 222

QUERY_MDS_STATISTICS . . . 225
QUERY_MODE . . . 228
QUERY_MODE_DEFINITION . . . 234
QUERY_MODE_TO_COS_MAPPING . . . 238
QUERY_NMVT_APPLICATION . . . 241
QUERY_NODE . . . 244
QUERY_PARTNER_LU . . . 252
QUERY_PARTNER_LU_DEFINITION . . . 258
QUERY_PORT . . . 263
QUERY_PU . . . 271
QUERY_RTP_CONNECTION . . . 276
QUERY_SESSION . . . 283
QUERY_STATISTICS . . . 290
QUERY_TP . . . 292
QUERY_TP_DEFINITION . . . 296

Chapter 7. Session Limit Verbs . . . 301
CHANGE_SESSION_LIMIT . . . 302
INITIALIZE_SESSION_LIMIT . . . 306
RESET_SESSION_LIMIT . . . 310

Chapter 8. Node Operator Facility API Indications . . . 313
DLC_INDICATION . . . 314
DLUS_INDICATION . . . 315
FOCAL_POINT_INDICATION . . . 317
LOCAL_TOPOLOGY_INDICATION . . . 319
LS_INDICATION . . . 320
LU_0_TO_3_INDICATION . . . 324
MODE_INDICATION . . . 328
PLU_INDICATION . . . 330
PORT_INDICATION . . . 332
PU_INDICATION . . . 333
REGISTRATION_FAILURE . . . 336
RTP_INDICATION . . . 337
SESSION_INDICATION . . . 341

Chapter 9. Security Verbs . . . 345
DEFINE_LU_LU_PASSWORD . . . 346
DEFINE_USERID_PASSWORD . . . 348
DELETE_LU_LU_PASSWORD . . . 350
DELETE_USERID_PASSWORD . . . 352

Chapter 10. APING and CPI-C Verbs . . . 355
APING . . . 356
CPI-C Verbs . . . 360
DEFINE_CPIC_SIDE_INFO . . . 361
DELETE_CPIC_SIDE_INFO . . . 364
QUERY_CPIC_SIDE_INFO . . . 366

Chapter 11. Attach Manager Verbs . . . 369
DISABLE_ATTACH_MANAGER . . . 370
ENABLE_ATTACH_MANAGER . . . 371
QUERY_ATTACH_MANAGER . . . 372

Part 2. Personal Communications Management Services API 373

Chapter 12. Introduction to Management Services API 375
Management Services Verbs 375
Entry Points 375
Verb Control Blocks (VCB) 375
Compiling and Linking Your Management Services Program 376
DLL Interface 376
Using Personal Communications High-Level Language Header Files . . . 377
Common VCB Fields 377

Chapter 13. Management Services Entry Points 379
WinMS() 380
WinMSCleanup() 381
WinMSStartup() 382
WinMSRegisterApplication() 383
WinMSUnregisterApplication() 385
WinMSGetIndication() 387

Chapter 14. Management Services Verbs 389
TRANSFER_MS_DATA 390
MDS_MU_RECEIVED 393
SEND_MDS_MU 395
ALERT_INDICATION 398
FP_INDICATION 399
NMVT_RECEIVED 400

Appendix A. IBM APPN MIB Tables 403

Glossary 405

Index 443

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make them available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594
USA

For license inquires regarding double-byte character set (DBCS) information, contact:

IBM World Trade Asia Corporation Commercial Relations
IBM Kamiya-cho Building, 4-3-9 Toranomon, Minato-ku
Tokyo, Japan

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, NC
27709-2195
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement.

This document is not intended for production use and is furnished as is without any warranty of any kind, and all warranties are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

© Copyright IBM Corp. 1989, 1996

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

Advanced Peer-to-Peer Networking
Application System/400
APPN
AS/400
CICS
IBM
System/370
VTAM

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

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

C-bus is a trademark of Corollary, Inc.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

About This Book

This book describes how to develop programs that use IBM Personal Communications AS/400 for Windows 95 (hereafter called PC400) and IBM Personal Communications AS/400 and 3270 for Windows 95. In this book, *Windows* refers to Windows 95. PC/3270 refers to the 3270 portion of the combined packaged. Throughout this book, workstation refers to all supported personal computers. When only one model or architecture of the personal computer is referred to, only that type is specified.

Who Should Read This Book

This book is intended for programmers and developers who plan on using Node Operator Facility (NOF) API messages to manage and query the operation of Personal Communications.

This book is also intended for developers who are writing network management applications that use the underlying Management Services support provided by Personal Communications to communicate with remote (host focal point) network management applications.

How to Use This Book

This book is organized into two parts. Part 1, “Personal Communications Node Operator Facility” describes the following:

Chapter 1, “Introduction,” describes the purpose of this book.

Chapter 2, “Overview,” describes the Node Operator Facility API structure and the verbs it supports. The chapter outlines the categories of the verbs implemented and the additional signals provided by Personal Communications.

Chapter 3, “Node Operator Facility Entry Points,” describes the entry point extensions.

Chapter 4, “Node Configuration Verbs” through Chapter 11, “Attach Manager Verbs,” describe the syntax of each verb. A copy of the structure which holds the information for each verb is included and each entry described, followed by a list of possible return codes.

Part 2, “Personal Communications Management Services API” describes the following:

Chapter 12, “Introduction to Management Services API,” describes the Management Services API.

Chapter 14, “Management Services Verbs,” describes the syntax of each verb. A copy of the structure which holds the information for each verb is included and each entry described, followed by a list of possible return codes.

© Copyright IBM Corp. 1989, 1996

Conventions Used in This Book

The following conventions are used throughout the Personal Communications library. Some of the conventions listed might not be used in this particular book.

Text Conventions

Bold	Bold type indicates verbs, functions, and parameters that you can use in a program or at a command prompt. These values are case sensitive and should be entered exactly as they appear in the text.
Italics	Italic type indicates the following things: A variable that you supply a value for. The names of window controls, such as lists, check boxed, entry fields, push buttons, and menu choices. They appear in the text as they appear in the window. Book titles. A letter is being used as a letter or a word is being used as a word. **Example:** When you see an *a*, make sure it is not supposed to be an *an*.
Bold italics	Bold italic type is used to emphasize a word.
UPPERCASE	Uppercase indicates constants, file names, keywords, and options that you can use in a program or at a command prompt. You can enter these values in uppercase or lowercase.
Double quotation marks	Double quotation marks indicate messages you see in a window. An example of this would be the messages that appear in the operator information area (OIA) of an emulator session.
Example type	Example type indicates information that you are instructed to type at a command prompt or in a window.

Number Conventions

Binary numbers	Represented as BX'xxxx xxxx' or BX'x' except in certain instances where they are represented with text ("A value of binary xxxx xxxx is...").
Bit positions	Start with 0 at the rightmost position (least significant bit).
Decimal numbers	Decimal numbers over 4 digits are represented in metric style. A space is used rather than a comma to separate groups of 3 digits. For example, the number sixteen thousand, one hundred forty-seven is written 16 147.
Hexadecimal numbers	Represented in text as hex xxxx or X'xxxx' ("The address of the adjacent node is hex 5D, which is specified as X'5d'.")

Double-Byte Character Set Support

Personal Communications supports double-byte character sets (DBCS), in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, displaying, and printing of DBCS characters require hardware and programs that support DBCS.

Where information applies specifically to DBCS, it is noted in this information unit.

ASCII refers to PC single-byte code in this book. ASCII should be considered as JISCII in Japan.

Where to Find More Information

Related Publications

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

IBM Systems Network Architecture: LU6.2 Reference: Peer Protocols, SC31-6808

IBM Systems Network Architecture: APPN Architecture Reference, SC30-3422

IBM Systems Network Architecture: Management Services, SC30-3346

IBM Systems Network Architecture: Formats, GA27-3136

IBM APPN Architecture and Product Implementations Tutorial, GG24-3669

IBM AS/400 Advanced Peer-to-Peer Networking, GG24-3287

IBM Communications Manager/2 System Management Programming Reference, SC31-6173

IBM Communications Manager/2 APPC Programming Guide and Reference, SC31-6160

IBM System/370 Principles of Operation, GA22-7000

IBM Systems Network Architecture: Technical Overview, GC30-3073

IBM Systems Network Architecture: VTAM Programming for LU Type 6.2, SC30-3400

IBM Systems Network Architecture Concepts and Products, GC30-3072

IBM Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2, SC30-3269

IBM Systems Network Architecture: Introduction to APPC, GG24-1584

IBM Systems Network Architecture: Transaction Programmer's Reference Manual for LU Type 6.2, GC30-3084

IBM Systems Network Architecture: Introduction to Sessions between Logical Units, GC20-1869

IBM Systems Network Architecture Format and Protocol Reference Manual: Architectural Logic, SC30-3112

Part 1. Personal Communications Node Operator Facility

Chapter 1. Introduction 5
Purpose of the Document 5
Personal Communications Node Operator Facility 5
Entry Points 5
Verb Control Blocks (VCB) 6
Compiling and Linking Your Node Operator Facility Program 6
DLL Interface 7
Using Personal Communications High-Level Language Header Files 7

Chapter 2. Overview 9
Overview of the Verbs in This Book 9
How To Read Verb Descriptions 9
Supplied Parameters 9
Returned Parameters 9
Common VCB Fields 10
Configuring the Node for LEN Node support 15
DLC Processes, Ports, and Link Stations 15

Chapter 3. Node Operator Facility Entry Points 17
WinNOF() 18
WinAsyncNOF() 19
WinAsyncNOFEx() 20
WinNOFCancelAsyncRequest() 21
WinNOFCleanup() 22
WinNOFStartup() 23
WinNOFRegisterIndicationSink() 24
WinNOFUnregisterIndicationSink() 25
WinNOFGetIndication() 26

Chapter 4. Node Configuration Verbs 27
DEFINE_ADJACENT_LEN_NODE 28
DEFINE_CN 31
DEFINE_COS 35
DEFINE_DEFAULTS 41
DEFINE_DEFAULT_PU 43
DEFINE_DLC 45
DEFINE_DLUR_DEFAULTS 48
DEFINE_FOCAL_POINT 50
DEFINE_INTERNAL_PU 54
DEFINE_LOCAL_LU 57
DEFINE_LS 60
DEFINE_LU_0_TO_3 70
DEFINE_MODE 73
DEFINE_PARTNER_LU 76
DEFINE_PORT 79
DEFINE_TP 85
DELETE_ADJACENT_LEN_NODE 88
DELETE_CN 90
DELETE_COS 92
DELETE_DLC 94

© Copyright IBM Corp. 1989, 1996

DELETE_FOCAL_POINT 96
DELETE_INTERNAL_PU 98
DELETE_LOCAL_LU 100
DELETE_LS 102
DELETE_LU_0_TO_3 104
DELETE_MODE 106
DELETE_PARTNER_LU 108
DELETE_PORT 110
DELETE_TP 112

Chapter 5. Activation and Deactivation Verbs 115
START_DLC 116
START_INTERNAL_PU 118
START_LS 121
START_PORT 123
STOP_DLC 125
STOP_INTERNAL_PU 127
STOP_LS 129
STOP_PORT 131
ACTIVATE_SESSION 133
DEACTIVATE_CONV_GROUP 136
DEACTIVATE_SESSION 138
PATH_SWITCH 141

Chapter 6. Query Verbs 143
QUERY_CN 144
QUERY_CN_PORT 149
QUERY_COS 152
QUERY_DEFAULT_PU 156
QUERY_DEFAULTS 158
QUERY_DIRECTORY_LU 160
QUERY_DIRECTORY_STATS 164
QUERY_DLC 166
QUERY_DLUR_LU 171
QUERY_DLUR_PU 175
QUERY_DLUS 181
QUERY_FOCAL_POINT 186
QUERY_LOCAL_LU 191
QUERY_LOCAL_TOPOLOGY 196
QUERY_LS 201
QUERY_LU_0_TO_3 214
QUERY_MDS_APPLICATION 222
QUERY_MDS_STATISTICS 225
QUERY_MODE 228
QUERY_MODE_DEFINITION 234
QUERY_MODE_TO_COS_MAPPING 238
QUERY_NMVT_APPLICATION 241
QUERY_NODE 244
QUERY_PARTNER_LU 252
QUERY_PARTNER_LU_DEFINITION 258
QUERY_PORT 263
QUERY_PU 271
QUERY_RTP_CONNECTION 276
QUERY_SESSION 283

QUERY_STATISTICS 290
QUERY_TP 292
QUERY_TP_DEFINITION 296

Chapter 7. Session Limit Verbs 301
CHANGE_SESSION_LIMIT 302
INITIALIZE_SESSION_LIMIT 306
RESET_SESSION_LIMIT 310

Chapter 8. Node Operator Facility API Indications 313
DLC_INDICATION 314
DLUS_INDICATION 315
FOCAL_POINT_INDICATION 317
LOCAL_TOPOLOGY_INDICATION 319
LS_INDICATION 320
LU_0_TO_3_INDICATION 324
MODE_INDICATION 328
PLU_INDICATION 330
PORT_INDICATION 332
PU_INDICATION 333
REGISTRATION_FAILURE 336
RTP_INDICATION 337
SESSION_INDICATION 341

Chapter 9. Security Verbs 345
DEFINE_LU_LU_PASSWORD 346
DEFINE_USERID_PASSWORD 348
DELETE_LU_LU_PASSWORD 350
DELETE_USERID_PASSWORD 352

Chapter 10. APING and CPI-C Verbs 355
APING 356
CPI-C Verbs 360
DEFINE_CPIC_SIDE_INFO 361
DELETE_CPIC_SIDE_INFO 364
QUERY_CPIC_SIDE_INFO 366

Chapter 11. Attach Manager Verbs 369
DISABLE_ATTACH_MANAGER 370
ENABLE_ATTACH_MANAGER 371
QUERY_ATTACH_MANAGER 372

Chapter 1. Introduction

This book describes the Node Operator Facility API provided by Personal Communications.

Purpose of the Document

The aim of the document is to do the following:

Provide a brief overview of the structure of the Node Operator Facility API
Define the syntax of the signals which flow across the interface.

Personal Communications Node Operator Facility

The Personal Communications Node Operator Facility enables communication between the node operator, and the Control Point (CP) and Logical Units (LUs). The Node Operator Facility receives node configuration information from the operator which it uses to initialize the CP when the node is started. The Node Operator Facility also receives requests to query and display node configuration information. The node operator is able to do the following:

Define and delete LUs, DLCs, ports, and links
Activate and deactivate links and sessions
Query the CP and LUs for data base and status information

The node operator can be a human operator working with an interactive display, a command file accessed by a file interface, or a transaction program. The Node Operator Facility communicates with the node operator via a message passing interface.

Entry Points

Personal Communications provides a library file that handles Node Operator Facility verbs.

Node Operator Facility verbs have a straightforward language interface. Your program fills in fields in a block of memory called a *verb control block* Then your program calls the entrypoint and passes a pointer to the verb control block. When its operation is complete, Node Operator Facility returns, having used and then modified the fields in the verb control block. Your program can then read the returned parameters from the verb control block.

The following is a list of entry points for Node Operator Facility verbs:

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

© Copyright IBM Corp. 1989, 1996

See Chapter 3, “Node Operator Facility Entry Points” for detailed descriptions of the entry points.

Verb Control Blocks (VCB)

Programming Note: The base operating system optimizes performance by executing some subsystems in the calling application's address space. This means that incorrect use of local descriptor table (LDT) selectors by application programs that have not been fully or correctly debugged can cause improper operation, or perhaps system failures. Accordingly, application programs should not perform pointer arithmetic operations that involve changing the LDT selector field of a pointer.

The segment used for the verb control block (VCB) must be a read/write data segment. Your program can either declare the VCB as a variable in your program, allocate it or suballocate it from a larger segment. It must be sufficiently large to contain all the fields for the verb your program is issuing.

An application program should not change any part of the verb control block after it has been issued until the verb completes. When Node Operator Facility finishes the execution of a verb, it copies a complete, modified VCB back onto the original block. Therefore, if your program declares a verb control block as a variable, consider declaring it in static storage rather than on the stack of an internal procedure.

Fill all reserved and unused fields in each VCB with zeros (X'00'). In fact, it might be more time-efficient to set the entire verb control block to zeros before your program assigns the values to the parameters. Setting reserved fields to zeros is particularly important.

Note: If the VCB is not read/write, or if it is not at least 10 bytes (that is, large enough to hold the Node Operator Facility primary and secondary return codes), Node Operator Facility cannot access it, and the base operating system abnormally ends the process. This termination is recognized as a *general protection fault*, processor exception trap D.

Node Operator Facility returns the INVALID_VERB_SEGMENT primary return code when the VCB is too short or the incorrect type of segment is used.

Compiling and Linking Your Node Operator Facility Program

The WINNOF32.LIB file can be found in the LIB subdirectory of the Personal Communications directory.

Save the source files for all the elements of your program. Later releases of Personal Communications might require that the program be recompiled and relinked to use the new functions in those releases.

You can create a program module (an .EXE file) to run on the base operating system by linking your compiled source files with the LINK utility program supplied with the base operating system. For more information on how to create a .EXE file, refer to the programming language reference manual associated with the language you are using.

The LINK program uses the compiled source files, a list of library files, and module definition files that contain information about your program. From these files, the LINK program builds the .EXE file that runs your program.

DLL Interface

Windows 95 provides two ways to dynamically link to a DLL:

Load-time linking
Run-time linking

To link to WINNOF32.DLL at load time, you should link with C:\Personal_Communications\lib\WINNOF32.LIB.

The WINNOF32.LIB library contains entry point linkage information for all Personal Communications API entry points.

Run-time linking provides conditional linking to a DLL. In order to find the function address of the NOF API entry point, your application can issue **LoadLibrary** for the module name of "WINNOF32" and then **GetProcAddress** for the address of the routines for the 32-bit NOF interface.

An application can use **FreeLibrary** to unlink after the application has finished using the NOF interface.

Using Personal Communications High-Level Language Header Files

The C language is supported. Only 32 bit API support is provided.

To simplify the coding of Node Operator Facility verbs, the VCB structure for each verb, along with appropriate parameter constants, is supplied with Personal Communications. The pseudonym file WINNOF.H is provided to assist in the development of programs. It contains the constant declarations and data types for each supplied and returned parameter in the NOF calls, and call prototypes. This file can be found in the INCLUDE subdirectory of the Personal Communications directory.

Chapter 2. Overview

Overview of the Verbs in This Book

The verb interface described in this book allows your programs to perform most of the configuration, system management functions, and node definition functions associated with a Personal Communications network environment. This section provides you with an overview for each of these functions and the associated verbs.

How To Read Verb Descriptions

Chapter 4, “Node Configuration Verbs” through Chapter 11, “Attach Manager Verbs” of this book contain descriptive listings of the configuration, system management, and attach manager verbs.

Supplied Parameters

Each verb description has a section that provides a detailed description of the parameters, and any associated parameter values, supplied by the program.

In some cases, you must supply a variable value for a parameter.

Returned Parameters

Each verb description has a section that provides a detailed description of the parameters and any associated parameter values returned to the program.

Return Codes

The configuration, system management, and attach manager verbs described in this book have return codes associated with them that supply information about the success of verb execution or that provide error information. These codes are listed in the “Returned Parameters” section for each verb.

Additional Information

Many of the verb descriptions also contain a section titled “Additional Information.” This section provides additional useful information about the verb.

© Copyright IBM Corp. 1989, 1996

Common VCB Fields

This chapter documents the syntax of each verb passed across the Node Operator Facility API. It also describes the parameters passed in and returned for each verb.

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

Each VCB has a number of common fields. These are listed and described below.

opcode Verb operation code. This field identifies the verb.

format Identifies the format of the VCB. The value that this field must be set to in order to specify the current version of the VCB is documented individually under each verb.

primary_rc Primary return code. Possible values for each verb are listed in the following sections.

secondary_rc Secondary return code. This supplements the information provided by the primary return code.

Verb Summary

The Node Operator Facility API is composed of verbs which can be used to do the following:

Configure node resources
Activate and deactivate links and sessions
Query information held by the node
Change the number of sessions
Handle unsolicited indications
Provide password support
“ping” a remote LU
Define, query, and delete CPI-C side information

Node Configuration

The following verbs can be used to define resources:

DEFINE_ADJACENT_LEN_NODE
DEFINE_CN
DEFINE_COS
DEFINE_DEFAULT_PU
DEFINE_DLC
DEFINE_DLUR_DEFAULTS
DEFINE_FOCAL_POINT
DEFINE_INTERNAL_PU
DEFINE_LOCAL_LU
DEFINE_LS
DEFINE_LU_0_TO_3
DEFINE_LU_0_TO_3_RANGE
DEFINE_LU_POOL

DEFINE_MODE
DEFINE_PARTNER_LU
DEFINE_PORT
DEFINE_TP

The following verbs can be used to delete resources:

DELETE_ADJACENT_LEN_NODE
DELETE_CN
DELETE_COS
DELETE_DLC
DELETE_FOCAL_POINT
DELETE_INTERNAL_PU
DELETE_LOCAL_LU
DELETE_LS
DELETE_LU_0_TO_3
DELETE_LU_0_TO_3_RANGE
DELETE_LU_POOL
DELETE_MODE
DELETE_PARTNER_LU
DELETE_PORT
DELETE_TP

Activation and Deactivation

The following verbs are used at link level:

START_DLC
START_INTERNAL_PU
START_LS
START_PORT
STOP_DLC
STOP_INTERNAL_PU
STOP_LS
STOP_PORT

The following verbs are used at session level:

ACTIVATE_SESSION
DEACTIVATE_CONV_GROUP
DEACTIVATE_SESSION

The following verb is used to force an RTP connection to switch paths:

PATH_SWITCH

Querying the Node

These verbs return node information in named fields:

QUERY_DEFAULT_PU
QUERY_DIRECTORY_STATS
QUERY_MDS_STATISTICS
QUERY_NODE
QUERY_STATISTICS

The following verbs may return one or more units of information:

QUERY_CN
QUERY_CN_PORT
QUERY_COS
QUERY_DEFAULTS
QUERY_DLUS
QUERY_FOCAL_POINT
QUERY_LU_POOL
QUERY_MDS_APPLICATION
QUERY_MODE_TO_COS_MAPPING
QUERY_NMVT_APPLICATION
QUERY_PU
QUERY_TP

This information can be thought of as being stored in the form of a list. The verb may specify a named entry in the list, which is then considered to be a place marker (or index value) in the list. The **list_options** field on these verbs specifies from which point in the list information will be returned.

If **list_options** is set to AP_FIRST_IN_LIST then the fields specifying the index value will be ignored, and the returned list will start at the beginning.

If **list_options** is set to AP_LIST_INCLUSIVE then the returned list will start from the specified index value.

If **list_options** is set to AP_LIST_FROM_NEXT then the returned list will start from the entry after the specified index value.

The index value specifies the starting point for returned information. Once this has been determined, some of the query verbs also provide additional filtering options for the returned list. These are specified independently of the index value. Note that unless specified otherwise, the returned list will be ordered according to IBM's 6611 APPN MIB. (See Appendix A, “IBM APPN MIB Tables,” for information on how verb parameters map to MIB table entries.)

The number of entries to be returned or the buffer size to be filled is set. (If both are set then the verb is returned with the lowest of the two specified quantities of information.) Because the application buffer size typically limits the amount of information that can be returned, the Node Operator Facility returns additional information indicating the total amount of buffer space required to return the requested information, and the total number of entries this represents.

In addition to returning one or more units of information, the following verbs are also able to return different levels of information. The **list_options** field specifies whether summary or detailed information will be returned by including either AP_DETAIL or AP_SUMMARY in the **list_options** field. These options are specified by **ORing** one of the previous **list_options**, for example: AP_DETAIL | AP_FIRST_IN_LIST.

QUERY_DIRECTORY_LU
QUERY_DLC
QUERY_DLUR_LU
QUERY_DLUR_PU
QUERY_LOCAL_LU
QUERY_LOCAL_TOPOLOGY
QUERY_LS
QUERY_LU_0_TO_3
QUERY_MODE
QUERY_MODE_DEFINITION
QUERY_PARTNER_LU
QUERY_PARTNER_LU_DEFINITION
QUERY_PORT
QUERY_RTP_CONNECTION
QUERY_SESSION
QUERY_TP_DEFINITION

Changing the number of sessions

CHANGE_SESSION_LIMIT
INITIALIZE_SESSION_LIMIT
RESET_SESSION_LIMIT

Unsolicited Indications

Applications displaying node information can use these indications (which are issued when a change occurs and return summary information only) to trigger the issue of query verbs (returning detailed information). The node only produces the signals listed below as unsolicited indications of the named events if there are any applications registered to receive the information. Applications should therefore unregister if they no longer require the information.

DLC_INDICATION
DLUS_INDICATION
FOCAL_POINT_INDICATION
LOCAL_TOPOLOGY_INDICATION
LS_INDICATION
LU_0_TO_3_INDICATION
MODE_INDICATION
PLU_INDICATION
PORT_INDICATION
PU_INDICATION
REGISTRATION_FAILURE
RTP_INDICATION
SESSION_INDICATION

The entry points used for indications are:

WinNOFRegisterIndicationSink	Register to receive an indication
WinNOFUnregisterIndicationSink	Unregister from receiving an indication
WinNOFGetIndication	Receive an indication

These indications are passed to any indication sinks that have registered with the Node Operator Facility. If the component generating the indication is unable to send it, then it sets the **data_lost** indicator on the next indication it issues. If the **data_lost** flag has been set to AP_YES on an indication then subsequent data fields may be set to null. This flag is used to notify the application that a change has occurred whose details have been lost, indicating that the application should respond by issuing the appropriate query verb.

Security Verbs

The following security verbs are modelled on verbs described in IBM's *Communications Manager/2 System Management Programming Reference*.

DEFINE_LU_LU_PASSWORD
DEFINE_USERID_PASSWORD
DELETE_LU_LU_PASSWORD
DELETE_USERID_PASSWORD

APING Verbs

The following verb allows a management application to “ping” a remote LU in the network.

APING

CPI-C Verbs

The following verbs allow CPI-C side information to be defined, queried, and deleted.

DEFINE_CPIC_SIDE_INFO
DELETE_CPIC_SIDE_INFO
QUERY_CPIC_SIDE_INFO

See the *CPI-C Reference* for more information about the CPI-C support provided by Personal Communications.

Attach Manager Verbs

The following verbs can be used to control the attach manager:

DISABLE_ATTACH_MANAGER
ENABLE_ATTACH_MANAGER
QUERY_ATTACH_MANAGER

Configuring the Node for LEN Node support

In order to support adjacent LEN nodes, Personal Communications must be configured as follows:

1. Personal Communications supports links to most LEN nodes when the **adj_cp_type** on the DEFINE_LS is set to AP_APPN_NODE. However, if the adjacent LEN node does not send the Network Node Control Vector in its XID3 (or if it is not clear what level of XID function the LEN node supports), then:

   Set the **adj_cp_type** field in the **def_data** structure of DEFINE_LS to AP_BACK_LEVEL_LEN_NODE

   Fill in the **adj_cp_name** field in the **def_data** structure of DEFINE_LS

   If the port is switched, then the **adj_node_id** field in the **def_data** structure of DEFINE_LS should be set to a unique four-byte hexadecimal string, to ensure that the identity of the adjacent node is checked by Personal Communications.

2. Define the resources on the LEN node using the DEFINE_ADJACENT_LEN_NODE verb.

DLC Processes, Ports, and Link Stations

DLC Processes

Personal Communications can create multiple DLC processes. Each DLC process is created by Personal Communications in response to a START_DLC verb issued at the Node Operator Facility API. Each DLC is responsible for communication over a link, or set of links, using a specific data link protocol (such as SDLC or Token Ring).

Each DLC process may manage one or more ports. Ports are described below.

Ports

A port represents a unique access point (such as a MAC/SAP address pair) in the local machine and is associated with a DLC process. Each DLC may have one or more ports. A port can be one of the following types:

Switched port Can have one or more adjacent link stations that are active at any one time. (Note that this differs from the definition in the *SNA APPN Architecture Reference*.)

Nonswitched port can have both point-to-point and multipoint link connections. Adjacent link stations on a nonswitched link connection must be defined by a Node Operator component. Multipoint nonswitched links require primary/secondary relationships to be defined properly on all nodes to avoid unpredictable results.

SATF port Uses a shared-access transport facility such as token ring. It allows connectivity between any pair of link stations attaching to the facility. The initial role for all link stations being activated on a token ring must always defined as negotiable so that link activation may be initiated through any link station.

Note: SATF Ports may also be associated with Connection Networks. In this case, topology updates are used to broadcast the address of the unique access point.

Link Stations

A link station is associated with a port and represents a connection to an adjacent node. A port can have multiple link stations. Link stations may be categorized in the following way:

Defined link station A link station that has been defined explicitly (using a DEFINE_LS verb).

Dynamic link station
A link station that has been created as a result of activating a dynamic connection through a Connection Network (also known as a Virtual Routing Node (VRN)).

Implicit link station A link station that has been created as a result of a call received from a previously unknown partner node on a switched or SATF port. (This type of port is not defined in the *SNA APPN Architecture Reference*.)

Temporary link station
A link station that is created when a CONNECT_IN is received over the DLC interface on a switched or SATF port. It is either deleted, or becomes dynamic or implicit, when the remote node identity is determined.

Chapter 3. Node Operator Facility Entry Points

This chapter describes the entry points for Node Operator Facility verbs.

© Copyright IBM Corp. 1989, 1996

WinNOF()

This function provides a synchronous entry point for all of the Node Operator Facility verbs.

Syntax

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

Parameter	Description
vcb	Pointer to verb control block.
vcb_size	Number of bytes in the verb control block.

Returns

No return value. The **primary_rc** and **secondary_rc** fields in the verb control block indicate any error.

Remarks

This is the main synchronous entry point for the Node Operator Facility API. This call blocks until the verb completes.

WinAsyncNOF()

This function provides an asynchronous entry point for all of the Node Operator Facility verbs.

Syntax

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

Parameter	Description
hwnd	Window handle to receive completion message.
vcb	Pointer to verb control block.
vcb_size	Number of bytes in the verb control block.

Returns

The return value specifies whether the asynchronous resolution request was successful. If the extension was successful, the actual return value is a handle. If the function was not successful, a zero is returned.

Remarks

Each application thread may only have one outstanding request at a time.

When the asynchronous operation is complete, the application's window *hWnd* receives the message returned **RegisterWindowMessage** with **"WinAsyncNOF"** as the input string. The *wParam* argument contains the asynchronous task handle returned by the original function call.

If the function returns successfully, a **WinAsyncNOF()** message will be posted to the application when the operation completes or the conversation is canceled.

Note: See also **WinNOFCancelAsyncRequest()**.

WinAsyncNOFEx()

This function provides an asynchronous entry point for all of the Node Operator Facility verbs. Use this extension instead of the blocking versions of the verbs to allow multiple sessions to be handled on the same thread.

Syntax

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

Parameter	Description
handle	Handle of event application will wait on.
vcb	Pointer to verb control block.
vcb_size	Number of bytes in the verb control block.

Returns

The return value specifies whether the asynchronous resolution request was successful. If the extension was successful, the actual return value is a handle.

Remarks

This verb is intended for use with WaitForMultipleObjects in the Win32 API. For more information about this function, see the programming documentation for the Win32 API.

Each application thread may only have one outstanding request at a time.

When the asynchronous operation is complete, the application is notified by way of the signaling of the event. Upon signaling of the event, examine the primary return code and secondary return code for any error conditions.

Note: See also **WinNOFCancelAsyncRequest()**.

WinNOFCancelAsyncRequest()

This function cancels an outstanding **WinAsyncNOF** based request.

Syntax

int WINAPI WinNOFCancelAsyncRequest(HANDLE handle);

Parameter	Description
handle	Supplied parameter; specifies the handle of the request to be canceled

Returns

The return value specifies whether the asynchronous request was canceled. If the value is zero, the request was canceled. Otherwise the value is:

WNOFALREADY
An error code indicating that the asynchronous request being canceled has already completed, or the handle was invalid.

Remarks

An asynchronous request previously issued by one of the **WinAsyncNOF** functions can be canceled prior to completion by issuing the **WinNOFCancelAsyncRequest()** call, specifying the handle returned by the initial function in *handle*.

Canceling an asynchronous request stops any update to the application verb control block and stops the application being notified that the verb has completed (either by way of the window message or event). It does not cancel the underlying request. To actually cancel the underlying request, the application must issue the appropriate NOF verb (that is, STOP_LS to cancel START_LS).

Should an attempt to cancel an existing asynchronous **WinAsyncNOF** routine fail with an error code of WNOFALREADY, one of two things has occurred. Either the original routine has already completed and the application has dealt with the resulting notification, or the original routine has already completed but the application has not dealt with the completion notification.

Note: See also **WinAsyncNOF()**.

WinNOFCleanup()

This function terminates and deregisters an application from the Node Operator Facility API.

Syntax

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

Returns

The return value specifies whether the deregistration was successful. If the value is not zero, the application was successfully deregistered. The application was not deregistered if a value of zero is returned.

Remarks

Use **WinNOFCleanup()** to indicate deregistration of a Node Operator Facility application from the Node Operator Facility API.

WinNOFCleanup will unblock any thread waiting in **WinNOFGetIndication**. These will return with WNOFNOTREG, (the application is not registered to receive indication). **WinNOFCleanup** will unregister the application for all indications. **WinNOFCleanup** will return any outstanding verb (synchronous or asynchronous) with the error AP_CANCELLED. However, the verb completes inside the node.

It is not a requirement to use **WinNOFStartup** and **WinNOFCleanup**. However, an application must be consistent in its use of these calls. You should use both of them or never use either of them. If a Node Operator Facility application does not call **WinNOFStartup** and **WinNOFCleanup** than it may not be able to operate successfully with later versions of Personal Communications.

Note: See also **WinNOFStartup()**.

WinNOFStartup()

This function allows an application to specify the version of Node Operator Facility API required and to retrieve the version of the API supported by the product. This function may be called by an application before issuing any further Node Operator Facility API calls to register itself.

Syntax

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

Parameter	Description
wVersionRequired	Specifies the version of Node Operator Facility API support required. The high-order byte specifies the minor version (revision) number; the low-order byte specifies the major version number.
nofdata	Returns version of Node Operator Facility API and description of API implementation.

Returns

The return value specifies whether the application was registered successfully and whether the Node Operator Facility API implementation can support the specified version number. If the value is zero, it was registered successfully and the specified version can be supported. Otherwise, the return value is one of the following:

WNOFSYSERROR
The underlying network subsystem is not ready for network communication.

WNOFVERNOTSUPPORTED
The version of Node Operator Facility API support requested is not provided by this particular implementation.

WNOFBADPOINTER
Invalid nofdata parameter.

Remarks

This call is intended to help with compatibility of future releases of the API. The current version is 1.0.

It is not a requirement to use **WinNOFStartup** and **WinNOFCleanup**. However, an application must be consistent in its use of these calls. You should use both of them or never use either of them. If a Node Operator Facility application does not call **WinNOFStartup** and **WinNOFCleanup** than it may not be able to operate successfully with later versions of Personal Communications.

Note: See also **WinNOFCleanup()**.

WinNOFRegisterIndicationSink()

This allows the application to register to receive unsolicited indications.

Syntax

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

Parameter	Description
indication_opcode	The indication to register for.
queue_size	Number of unreceived indications to queue. Zero means use current value (initial default value is set to ten). There is only one queue which is used for all indications registered by application.
primary_rc	Returned: primary return code
secondary_rc	Returned: secondary return code

Returns

The function returns a value indicating whether the registration was successful. If the value is not zero, the registration was successful. If the value is zero, the registration was not successful.

Remarks

Use **WinNOFRegisterIndicationSink** to register to receive unsolicited indications of type **indication_opcode**.

An application must issue a **WinNOFRegisterIndicationSink** for each type of indication it wants to receive.

Note: See also **WinNOFUnregisterIndicationSink** and **WinNOFGetIndication**.

WinNOFUnregisterIndicationSink()

This allows the application to stop receiving unsolicited indications.

Syntax

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

Parameter	Description
indication_opcode	The indication to unregister from.
primary_rc	Returned: primary return code.
secondary_rc	Returned: secondary return code.

Returns

The function returns a value indicating whether the unregistration was successful. If the value is not zero, the unregistration was successful. If the value is zero, the unregistration was not successful.

Remarks

Use **WinNOFUnregisterIndicationSink** to stop receiving unsolicited indications of type **indication_opcode**.

An application must issue a **WinNOFUnregisterIndicationSink** for each type of indication it wants to stop receiving.

Note: See also **WinNOFRegisterIndicationSink** and **WinNOFGetIndication**.

WinNOFGetIndication()

This allows the application to received unsolicited indications.

Syntax

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

Parameter	Description
buffer	Pointer to buffer to receive indication into.
buffer_size	Size of buffer. Returned: the size of the indication.
timeout	Time to wait for indication in milliseconds.

Returns

The function returns a value indicating whether an indication was received.

0 Indication returned.

WNOFTIMEOUT
Timeout waiting for indication.

WNOFSYSNOTREADY
The underlying network subsystem is not ready for network communication.

WNOFNOTREG
The application is not registered to receive indications.

WNOFBADSIZE
The buffer is too small to receive the indication. Re-issue the **WinNOFGetIndication** call with a large enough buffer. The size of the indication is returned in the **buffer_size** parameter.

WNOFBADPOINTER
Either the buffer or **buffer_size** parameter is invalid.

WNOFSYSERROR
An unexpected system error has occurred.

Remarks

This is a blocking call, it returns in one of the following circumstances:

An indication is returned
The timeout expires
The application issues a WinNOFCleanup call
The product is stopped
A system error occurs

Note: See also **WinNOFRegisterIndicationSink** and **WinNOFUnregisterIndicationSink**.

Chapter 4. Node Configuration Verbs

The following verbs are used to define and delete node configuration information.

© Copyright IBM Corp. 1989, 1996

DEFINE_ADJACENT_LEN_NODE

DEFINE_ADJACENT_LEN_NODE adds entries to the node directory database for the resources on an adjacent LEN Node.

This verb may be issued on an End Node, in which case the LEN Node's CP is added to the root of the directory

To define the LEN node's CP LU, set the following fields:

Specify the LEN node's CP name in the **cp_name** field

Add the CP name to the list of LUs specified in the **lu_names** array

Any additional LUs on the LEN Node are added to the directory as children of the LEN Node's CP. DEFINE_ADJACENT_LEN_NODE can also be used to add LU definitions to an existing LEN Node definition. Up to ten LUs can be added at a time. (LUs can be removed in the same way by issuing the DELETE_ADJACENT_LEN_NODE verb.) The LUs specified on a single verb must either be all wildcard entries, or all explicit entries. If the verb fails part way through processing, all new directory entries are removed, leaving the directory as it was before the verb was issued.

VCB Structure

```
typedef struct define_adjacent_len_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     num_of_lus;              /  number of LUs            /
        unsigned char     wildcard_lus;            /  wildcard LUs             /
        unsigned  char    reserv3[8];              /  reserved                 /
        unsigned char     lu_names[1 ][8];         /  LU names                 /
} DEFINE_ADJACENT_LEN_NODE;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_ADJACENT_LEN_NODE

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

cp_name The fully qualified name of the CP in the adjacent LEN end node. This should match the name the LEN Node sends on its XIDs (if it supports them), and the adjacent CP name specified on the DEFINE_LS for the link to the LEN Node. The name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

description Resource description (returned on QUERY_DIRECTORY_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

num_of_lus The number of LUs to be defined, in the range 0 to 10.

wildcard_lus Indicates whether the LU names are to be defined as wildcards (AP_YES or AP_NO).

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

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_CP_NAME
AP_INVALID_LU_NAME
AP_INVALID_NUM_LUS
AP_INVALID_WILDCARD_NAME

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_INVALID_CP_NAME
AP_INVALID_LU_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR
secondary_rc	AP_MEMORY_SHORTAGE AP_DIRECTORY_FULL

DEFINE_CN

DEFINE_CN defines a Connection Network (also known as a Virtual Routing Node or VRN). The verb provides the network-qualified name of the connection network along with its Transmission Group (TG) characteristics. It also provides is a list of the names of the local ports that can access this connection network.

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

VCB Structure

```
typedef struct define_cn
{
        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   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        /
        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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_CN

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

fqcn_name
Fully qualified name (17 bytes long) of connection network being defined. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

def_data.description
Resource description (returned on QUERY_CN). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_data.num_ports
Number of ports associated with this connection network. There may be up to and including 8 ports per DEFINE_CN verb, and up to and including 239 ports in total per CN.

def_data.tg_chars.effect_cap
Actual units of effective capacity. The value is encoded as a 1-byte floating point number, represented by the formula 0.1mmm * 2 eeeee where the bit representation of the byte is eeeeemmm. Each unit of effective capacity is equal to 300 bits per second.

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

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

def_data.tg_chars.security
Security values as described in the list below:

AP_SEC_NONSECURE
No security exists.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data transmitted over this connection network will flow over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data transmitted over secure underground cable.

AP_SEC_SECURE_CONDUIT
The line is a secure conduit that is not guarded.

AP_SEC_GUARDED_CONDUIT
Conduit is protected against physical tapping.

AP_SEC_ENCRYPTED
Encryption over the line.

AP_SEC_GUARDED_RADIATION
Line is protected against physical and radiation tapping.

def_data.tg_chars.prop_delay
Propagation delay representing the time it takes for a signal to travel the length of the link, in microseconds. The value is encoded as a 1-byte floating point number, represented by the formula 0.1mmm * 2 eeeee

where the bit representation of the byte is eeeeemmm. Default values are listed below:

AP_PROP_DELAY_MINIMUM
No propagation delay.

AP_PROP_DELAY_LAN
Less than 480 microseconds delay.

AP_PROP_DELAY_TELEPHONE
Between 480 and 49 512 microseconds delay.

AP_PROP_DELAY_PKT_SWITCHED_NET
Between 49 512 and 245 760 microseconds delay.

AP_PROP_DELAY_SATELLITE
Longer than 245 760 microseconds delay.

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

def_data.tg_chars.modem_class
Reserved. This field should always be set to zero.

def_data.tg_chars.user_def_parm_1
User defined parameter in range 0—255.

def_data.tg_chars.user_def_parm_2
User defined parameter in range 0—255.

def_data.tg_chars.user_def_parm_3
User defined parameter in range 0—255.

port_name
Array of up to eight port names defined on the connection network. Each named port must have already been defined by a DEFINE_PORT verb. Each port name is an 8-byte string in a locally displayable character set and must match that on the associated DEFINE_PORT verb. Additional ports may be defined on the Connection Network by issuing another DEFINE_CN specifying the new port names.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc	AP_OK

Returned Parameters: Parameter Check

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

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

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_PORT_ACTIVE

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_COS

DEFINE_COS adds a Class of Service definition. The DEFINE_COS verb can also be used to modify any fields in a previously defined COS.

The definition provides node and TG “rows.” These rows associate a range of node and TG characteristics with weights which are used for route calculation. The lower the weight the more favorable the route.

VCB Structure

The DEFINE_COS verb contains a variable number of **cos_tg_row** and **cos_node_row** overlays. The **cos_tg_row** structures are concatenated onto the end of DEFINE_COS (and ordered in ascending weight) and are followed by the **cos_node_row** structures (also ordered in ascending weight).

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_COS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

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

description
Resource description (returned on QUERY_COS). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

transmission_priority
Transmission priority. This is set to one of the following:

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

num_of_node_rows
Number of node row overlays which follow the DEFINE_COS VCB. The maximum is 8.

num_of_tg_rows
Number of TG row overlays which follow the node row overlays. The maximum is 8. Each node row contains a set of minimum node characteristics, a set of maximum node characteristics, and a weight. When computing the weights for a node, its characteristics are checked against the minimum and maximum characteristics defined for each node row. The node is then assigned the weight of the first node row which bounds all the node's characteristics within the limits specified. If the node characteristics do not satisfy any of the listed node rows, the node is considered unsuitable for this COS, and is assigned an infinite weight. Note that the node rows must be concatenated in ascending order of weight.

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

cos_node_row.minimum.status
Specifies the minimum congestion status of the node. This can be one of the following values:

AP_UNCONGESTED
The node is not congested.

AP_CONGESTED
The number of ISR sessions is greater than the **isr_sessions_upper_threshold** specified on the START_NODE verb.

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

cos_node_row.maximum.status
Specifies the maximum congestion status of the node. This can be one of the following values:

AP_UNCONGESTED
The node is not congested.

AP_CONGESTED
The number of ISR sessions is greater than the **isr_sessions_upper_threshold** specified on the START_NODE verb.

cos_node_row.weight
Weight associated with this node row. Values must be in the range 0—255. Each TG row contains a set of minimum TG characteristics, a set of maximum TG characteristics, and a weight. When computing the weights for a TG, its characteristics are checked against the minimum and maximum characteristics defined for each TG row. The TG is then assigned the weight of the first TG row which bounds all the TG's characteristics within the limits specified. If the TG characteristics do not satisfy any of the listed TG rows, the TG is considered unsuitable for this COS, and is assigned an infinite weight. Note that the TG rows must be concatenated in ascending order of weight.

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

cos_tg_row.minimum.connect_cost
Minimum limit for cost per connect time. Valid values are integer values in the range 0—255, where 0 is the lowest cost per connect time and 255 is the highest.

cos_tg_row.minimum.byte_cost
Minimum limit for cost per byte. Valid values are integer values in the range 0—255, where 0 is the lowest cost per byte and 255 is the highest.

cos_tg_row.minimum.security
Minimum limits for security values as described in the list below:

AP_SEC_NONSECURE
No security exists.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data transmitted over this connection network will flow over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data transmitted over secure underground cable.

AP_SEC_SECURE_CONDUIT
The line is a secure conduit that is not guarded.

AP_SEC_GUARDED_CONDUIT
Conduit is protected against physical tapping.

AP_SEC_ENCRYPTED
Encryption over the line.

AP_SEC_GUARDED_RADIATION
Line is protected against physical and radiation tapping.

cos_tg_row.minimum.prop_delay
Minimum limits for propagation delay representing the time it takes for a signal to travel the length of the link, in microseconds. The value is encoded as a 1-byte floating point number, represented by the formula 0.1mmm * 2 eeeee where the bit representation of the byte is eeeeemmm. Default values are listed below:

AP_PROP_DELAY_MINIMUM
No propagation delay.

AP_PROP_DELAY_LAN
Less than 480 microseconds delay.

AP_PROP_DELAY_TELEPHONE
Between 480 and 49 512 microseconds delay.

AP_PROP_DELAY_PKT_SWITCHED_NET
Between 49 512 and 245 760 microseconds delay.

AP_PROP_DELAY_SATELLITE
Longer than 245 760 microseconds delay.

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

cos_tg_row.minimum.modem_class
Reserved. This field should always be set to zero.

cos_tg_row.minimum.user_def_parm_1
Minimum limit for user defined parameter in range 0—255.

cos_tg_row.minimum.user_def_parm_2
Minimum limit for user defined parameter in range 0—255.

cos_tg_row.minimum.user_def_parm_3
Minimum limit for user defined parameter in range 0—255.

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

cos_tg_row.maximum.connect_cost
Maximum limit for cost per connect time. Valid values are integer values in the range 0—255, where 0 is the lowest cost per connect time and 255 is the highest.

cos_tg_row.maximum.byte_cost
Maximum limit for cost per byte. Valid values are integer values in the range 0—255, where 0 is the lowest cost per byte and 255 is the highest.

cos_tg_row.maximum.security
Maximum limits for security values as described in the list below:

AP_SEC_NONSECURE
No security exists.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data transmitted over this connection network will flow over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data transmitted over secure underground cable.

AP_SEC_SECURE_CONDUIT
The line is a secure conduit that is not guarded.

AP_SEC_GUARDED_CONDUIT
Conduit that is protected against physical tapping.

AP_SEC_ENCRYPTED
Encryption over the line.

AP_SEC_GUARDED_RADIATION
Line is protected against physical and radiation tapping.

cos_tg_row.maximum.prop_delay
Maximum limits for propagation delay representing the time it takes for a signal to travel the length of the link, in microseconds. The value is encoded as a 1-byte floating point number, represented by the formula 0.1mmm * 2 eeeee where the bit representation of the byte is eeeeemmm. Default values are listed below:

AP_PROP_DELAY_MINIMUM
No propagation delay.

AP_PROP_DELAY_LAN
Less than 480 microseconds delay.

AP_PROP_DELAY_TELEPHONE
Between 480 and 49 512 microseconds delay.

AP_PROP_DELAY_PKT_SWITCHED_NET
Between 49 512 and 245 760 microseconds delay.

AP_PROP_DELAY_SATELLITE
Longer than 245 760 microseconds delay.

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

cos_tg_row.maximum.modem_class
Reserved. This field should always be set to zero.

cos_tg_row.maximum.user_def_parm_1
Maximum limit for user defined parameter in range 0—255.

cos_tg_row.maximum.user_def_parm_2
Maximum limit for user defined parameter in range 0—255.

cos_tg_row.maximum.user_def_parm_3
Maximum limit for user defined parameter in range 0—255.

cos_tg_row.weight
Weight associated with this TG row.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc	AP_OK

Returned Parameters: Parameter Check

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

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

Returned Parameters: State Check

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

primary_rc	AP_STATE_CHECK
secondary_rc	AP_COS_TABLE_FULL

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc	AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DEFAULTS

DEFINE_DEFAULTS allows the user to define or redefine default actions of the node.

VCB Structure

```
typedef struct define_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      /
} DEFINE_DEFAULTS;

typedef struct default_chars
{
        unsigned  char    description[RD_LEN];
                                           / resource description     /
        unsigned  char    mode_name[8];    / default mode name        /
        unsigned  char    reserv[248];     / reserved                 /
} DEFAULT_CHARS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_DEFAULTS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

default_chars.description
Resource description (returned on QUERY_DEFAULTS). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

default_chars.mode_name
Name of the mode that will serve as the default. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Mode Name Invalid

If the verb specifies a default mode that is not valid (for example, not EBCDIC A), or is valid but has not been defined, Personal Communications returns the following parameters:

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_MODE_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DEFAULT_PU

DEFINE_DEFAULT_PU allows the user to define, redefine, or modify any field of a default PU. If a PU name is not specified explicitly on a TRANSFER_MS_DATA verb, then the management services information carried on the TRANSFER_MS_DATA is sent on the default PU's session with the host SSCP. For more information about this see Chapter 14, “Management Services Verbs.”

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_DEFAULT_PU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

pu_name Name of local PU that will serve as the default. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

description Resource description (returned on QUERY_DEFAULT_PU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DLC

DEFINE_DLC defines a new DLC or modifies an existing DLC. This verb defines the DLC name, which is unique throughout the node, and some DLC-specific data, which is concatenated to the basic structure. This data is used during initialization of the DLC, and the format is specific to the DLC type (such as Token Ring). Only the DLC-specific data appended to the verb can be modified using the DEFINE_DLC verb. In order to do this, a STOP_DLC verb must first be issued so that the DLC is in a reset state.

See “DLC Processes, Ports, and Link Stations” on page 15, for more information about the relationship between DLCs, ports and link stations.

VCB Structure

```
typedef struct define_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                 /
        DLC_DEF_DATA     def_data;            / DLC defined data            /
} DEFINE_DLC;

typedef struct dlc_def_data
{
        unsigned  char   description[RD_LEN];
                                              / 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   reserv3[11];         / reserved                    /
        unsigned short   dlc_spec_data_len; / Length of DLC specific data /
} DLC_DEF_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_DLC

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

dlc_name
Name of the DLC. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. Valid values are LAN, SDLC or TWINAX (padded to 8 chars with spaces).

def_data.description
Resource description (returned on QUERY_DLC). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_data.dlc_type
Type of the DLC. Personal Communications supports the following types:

AP_LLC2
AP_SDLC
AP_ TWINAX

def_data.neg_ls_supp
Specifies whether the DLC supports negotiable link stations (AP_YES or AP_NO). If the **dlc_type** is AP_TWINAX, then only AP_NO is supported.

def_data.port_types
Specifies the allowable port types for the supplied **dlc_type**. The value corresponds to one or more of the following ORed together.

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

Use the following table to set the fields for the corresponding DLC type.

Table 1. Port types for DLC types

DLC Type	Port Type
AP_LLC2	AP_SATF
AP_SDLC	AP_SWITCHED or AP_NON_SWITCHED
AP_TWINAX	AP_NONSWITCHED

def_data.dlc_spec_data_len
This field should always be set to zero.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_DLC_NAME
AP_INVALID_DLC_TYPE
AP_INVALID_PORT_TYPE

Returned Parameters: State Check

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

primary_rc	AP_STATE_CHECK
secondary_rc	AP_DLC_ACTIVE AP_INVALID_DLC_TYPE

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc	AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DLUR_DEFAULTS

DEFINE_DLUR_DEFAULTS allows the user to define, redefine, or revoke a default Dependent LU Server (DLUS) and a backup default DLUS. The default DLUS name is used by DLUR when it initiates SSCP-PU activation for PUs that do not have an explicitly specified associated DLUS. If a DLUS name is not specified explicitly on the DEFINE_DLUR_DEFAULTS verb then the current default (or backup DLUS) is revoked.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_DLUR_DEFAULTS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

description
Resource description. This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

dlus_name
Name of DLUS node which will serve as the default. This should be set to all zeros or a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, which is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If this field is set to all zeros, the current default DLUS is revoked.

bkup_dlus_name
Name of DLUS node which will serve as the backup default. This should be set to all zeros or a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, which is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If this field is set to all zeros, the current backup default DLUS is revoked.

dlus_retry_timeout
Interval in seconds between second and subsequent attempts to contact a DLUS. The interval between the initial attempt and the first retry is always one second. If zero is specified, the default value of 5 seconds is used.

dlus_retry_limit
Maximum number of retries after an initial failure to contact a DLUS. If zero is specified, the default value of 3 is used. If X'FFFF' is specified, Personal Communications will retry indefinitely.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_DLUS_NAME
AP_INVALID_BKUP_DLUS_NAME

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_FOCAL_POINT

Personal Communications can have a number of types of relationships with different focal points. The DEFINE_FOCAL_POINT verb defines a focal point with which Personal Communications has an implicit relationship (which may be of type primary or backup). These relationships, and the ways in which they can be established, are described below: Relationships between an MS Focal Point (FP) and an MS Entry Point (EP) for a given category are established when they exchange MS Capabilities messages. The following types of FP-EP relationships can be established.

Explicit
This relationship is established by an operator at the FP assigning the EP to its sphere of control. The FP initiates exchange of MS Capabilities.

Implicit (primary)
This relationship is established when an operator at an EP assigns the EP to a specified FP (for example, when the operator issues a DEFINE_FOCAL_POINT verb). The EP initiates the MS Capabilities exchange.

Implicit (backup)
This relationship is established when an EP loses either an explicit or implicit primary focal point. The EP initiates MS Capabilities exchange. The identity of the backup FP may be defined (using the DEFINE_FOCAL_POINT verb) or may be acquired via MS Capabilities exchange.

Domain
This relationship is established when a serving network node (NN) informs the end node EP of the identity of the FP. Domain relationships are only valid in End Nodes.

Host
This relationship does not involve MS Capabilities exchange and is established by the configuration of an SSCP-PU session from the EP node to a host. It is the lowest precedence focal point relationship.

Each DEFINE_FOCAL_POINT verb can only be used to define an implicit focal point (which may be of type primary or backup). Each DEFINE_FOCAL_POINT verb is issued for a specific Management Services Category. Within this category the DEFINE_FOCAL_POINT verb can be used to

define a focal point
replace a focal point (or backup focal point)
revoke the currently active focal point.

The fields on a DEFINE_FOCAL_POINT verb are used in the following way.

The **ms_category** must always be filled in. The combination of the **fp_fqcp_name** and **ms_appl_name fields** specify the focal point (or backup focal point if the **backup** field is set to AP_YES) for the specified category.

If the verb is being issued to revoke the currently active focal point without providing a new one, the **fp_fqcp_name** and **ms_appl_name** fields should be set to all zeros. When a DEFINE_FOCAL_POINT verb defining or replacing a focal point is received, Personal Communications attempts to establish an implicit primary focal point relationship with the specified focal point by sending an MS Capabilities request. When Personal Communications receives a

DEFINE_FOCAL_POINT verb revoking the currently active focal point, it sends an MS Capabilities revoke message to the focal point. It is recommended that the DELETE_FOCAL_POINT verb (specifying AP_ACTIVE) be used to revoke the currently active focal point.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_FOCAL_POINT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

ms_category
Management Services category. This may either be one of the four byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services categories as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation-defined name.

fp_fqcp_name
Focal point's fully qualified control point name. This should be set to all zeros or a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the focal point is being revoked, this field should be set to all zeros.

ms_appl_name
Focal point application name. This may either be one of the four-byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services applications as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation-defined name. If the focal point is being revoked, this field should be set to all zeros.

description
Resource description (returned on QUERY_FOCAL_POINT). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

backup Specifies whether a backup focal point is being defined (AP_YES or AP_NO). This field is reserved if the currently active focal point is being revoked. It is recommended that the DELETE_FOCAL_POINT verb (specifying AP_ACTIVE) be used to revoke the currently active focal point.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_FP_NAME
AP_INVALID_CATEGORY_NAME

Returned Parameters: Other Conditions

If the verb does not execute successfully, Personal Communications returns the following parameters:

primary_rc AP_REPLACED
AP_UNSUCCESSFUL

secondary_rc AP_IMPLICIT_REQUEST_REJECTED
AP_IMPLICIT_REQUEST_FAILED

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

Personal Communications returns the following parameters if the verb does not execute because of a system error or because Personal Communications failed to contact the focal point successfully:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_INTERNAL_PU

The DEFINE_INTERNAL_PU verb defines a DLUR-served local PU. Note that the DEFINE_LS verb should be used to define:

a downstream PU served by

- DLUR
- PU concentration

a local PU which is directly attached to the host.

VCB Structure

```
typedef struct define_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         /
        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  char    reserv2[8];             / reserved                  /
} INTERNAL_PU_DEF_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_INTERNAL_PU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

pu_name Name of the internal PU that is being defined. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

def_data.description
Resource description (returned on QUERY_DLUR_PU and QUERY_PU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_data.dlus_name
Name of DLUS node which DLUR will use when it initiates SSCP-PU activation. This should be set to all zeros or a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the field is set to all zeros, the global default DLUS (if it has been defined, using

the DEFINE_DLUR_DEFAULTS verb) is used in DLUR-initiated SSCP-PU activation.

def_data.bkup_dlus_name
Name of DLUS node which will serve as the backup DLUS for this PU. This should be set to all zeros or a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the field is set to all zeros, the global backup default DLUS (if it has been defined by the DEFINE_DLUR_DEFAULTS verb) is used as the backup for this PU.

def_data.pu_id
PU identifier. This a 4-byte hexadecimal string. Bits 0—11 are set to the Block number and bits 12—31 to the ID number which uniquely identifies the PU. This must match the **pu_id** configured at the host.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc	AP_OK

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_PU_NAME AP_INVALID_PU_ID AP_INVALID_DLUS_NAME AP_INVALID_BKUP_DLUS_NAME

Returned Parameters: State Check

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

primary_rc	AP_STATE_CHECK
secondary_rc	AP_PU_ALREADY_DEFINED

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_LOCAL_LU

The DEFINE_LOCAL_LU verb requests the definition of a local LU with the specified characteristics, or, if the LU already exists, the modification of the **attach_routing_data** characteristic of the LU. Note that a DEFINE_LOCAL_LU used to modify an existing definition will be rejected if any parameter other than the **attach_routing_data** field has changed.

VCB Structure

```
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[RD_LEN];
                                              / 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   reserv1;             / reserved                     /
        unsigned  char   reserv2;             / reserved                     /
        unsigned char    pu_name[8];          / PU name                      /
        unsigned  char   reserv3[8];          / reserved                     /
        unsigned  char   attach_routing_data[128];
                                              / routing data for             /
                                              / incoming attaches            /
} LOCAL_LU_DEF_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_LOCAL_LU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name Name of the local LU that is being defined. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

def_data.description
Resource description (returned on QUERY_LOCAL_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_data.lu_alias
Alias of the local LU to define. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set.

def_data.nau_address
Network addressable unit address of the LU which must be in the range 0—2554. A nonzero value implies the LU is a dependent LU. Zero implies the LU is an independent LU.

def_data.syncpt_support
Reserved. This field should always be set to AP_NO.

def_data.lu_session_limit
Maximum number of sessions supported by the LU. Zero means no limit. If the LU is independent then this can be set to any value. If the LU is dependent then this must be set to one.

def_data.pu_name
Name of the PU which this LU will use. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is only used by dependent LUs, and should be set to all binary zeros for independent LUs.

def_data.attach_routing_data
Reserved.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_LU_NAME
AP_INVALID_NAU_ADDRESS
AP_INVALID_SESSION_LIMIT

Returned Parameters: State Check

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

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

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

secondary_rc AP_MEMORY_SHORTAGE

DEFINE_LS

DEFINE_LS is used to define a new link station (LS) or modify an existing one. This verb provides the LS name, which is unique throughout the node, and the name of the port this LS should use. This port must already have been defined using a DEFINE_PORT verb. Link specific data is concatenated to the basic structure. DEFINE_LS can only be used to modify one or more fields of an existing link station if the link station is in a reset state (after a STOP_LS has been issued), and the **port_name** specified on the DEFINE_LS has not changed since the previous definition of the LS.

See “DLC Processes, Ports, and Link Stations” on page 15, for more information about the relationship between DLCs, ports and link stations.

The setting of a large number of the fields in **LS_DEF_DATA** depends on the value of the **adj_cp_type** field.

There are six values that **adj_cp_type** can take (which are described further under **def_data.adj_cp_type**), four of which are used for links to adjacent Type 2.1 (APPN) nodes:

AP_NETWORK_NODE
AP_END_NODE
AP_APPN_NODE
AP_BACK_LEVEL_LEN_NODE

and two of which are used for links carrying PU Type 2.0 traffic only:

AP_HOST_XID3
AP_HOST_XID0.

There are four types of APPN nodes which are distinguished as follows

An APPN network node includes the Network Name Control Vector (CV) in its XID3, supports parallel TGs, sets the networking capabilities bit in its XID3, and may support CP-CP sessions on a link.

An APPN end node includes the Network Name CV in its XID3, supports parallel TGs, does not set the networking capabilities bit in its XID3, and may support CP-CP sessions on a link.

An up-level LEN node includes the Network Name CV in its XID3, may support parallel TGs, does not set the networking capabilities bit in its XID3, and does not support CP-CP sessions.

A back-level LEN node does not include the Network Name CV in its XID3, does not support parallel TGs, does not set the networking capabilities bit in its XID3, and does not support CP-CP sessions.

The following fields must be set for all links:

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

Other fields must be set as follows:

If **adj_cp_type** is set to AP_NETWORK_NODE, AP_END_NODE, or AP_APPN_NODE the following fields must be set:

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

If **adj_cp_type** is set to AP_BACK_LEVEL_LEN_NODE the following fields must be set:

adj_cp_name
solicit_sscp_sessions
dspu_services
use_default_tg_chars
tg_chars

If a local PU is to use the link (**adj_cp_type** is set to AP_HOST_XID3 or AP_HOST_XID0, or **solicit_sscp_sessions** is set to AP_YES on a link to an APPN node) the following field must be set:

pu_name

VCB Structure

```
typedef struct define_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                /
        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   reserv3[35];             / 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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_LS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

ls_name Name of link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set.

def_data.description
Resource description (returned on QUERY_LS, QUERY_PU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_data.port_name
Name of port associated with this link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. This named port must have already been defined by a DEFINE_PORT verb.

def_data.adj_cp_name
Fully-qualified 17-byte adjacent CP name which is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is

only relevant for links to APPN nodes and is otherwise ignored. For links to APPN nodes it can be set to all zeros unless the field **tg_number** is set to a number in the range one to 20 or the field **adj_cp_type** is set to AP_BACK_LEVEL_LEN_NODE. If it is set to all zeros, it is not checked against the name received from the adjacent node during XID exchange. If it is not set to all zeros, it is checked against the name received from the adjacent node during XID exchange unless **adj_cp_type** is set to AP_BACK_LEVEL_LEN_NODE (in which case it is used to identify the adjacent node).

def_data.adj_cp_type
Adjacent node type.

AP_NETWORK_NODE
Specifies that the node is an APPN network node.

AP_END_NODE
Specifies that the node is an APPN end node or an up-level LEN node.

AP_APPN_NODE
Specifies that the node is an APPN network node, an APPN end node, or an up-level LEN node. The node type will be learned during XID exchange.

AP_BACK_LEVEL_LEN_NODE
Specifies that the node is a back-level LEN node.

AP_HOST_XID3
Specifies that the node is a host and that Personal Communications responds to a polling XID from the node with a format 3 XID.

def_data.dest_address.length
Length of destination link station's address on adjacent node.

def_data.dest_address.address
Link station's destination address on adjacent node.

def_data.auto_act_supp
Specifies whether the link can be activated automatically when required by a session. (AP_YES or AP_NO). If the link is not to an APPN node then this field can always be set to AP_YES and has no requirements on other parameters. If the link is to an APPN node then this field cannot be set to AP_YES if the link also supports CP-CP sessions (**cp_cp_sess_support** is set to AP_YES) and can only be set to AP_YES if a pre-assigned TG number is also defined for the link (**tg_number** is set to a value between one and 20). These requirements will always be met if **adj_cp_type** is set to AP_BACK_LEVEL_LEN_NODE since **cp_cp_sess_support** and **tg_number** are ignored in this case).

def_data.tg_number
Pre-assigned TG number. This field is only relevant if the link is to an adjacent APPN node and is otherwise ignored. If **adj_cp_type** is set to AP_BACK_LEVEL_LEN_NODE then it is also ignored and is assumed to be set to one. For links to adjacent APPN nodes this must be set in the range one to 20. This number is used to represent the link when the link is activated. Personal Communications will not accept any other number from the adjacent node during activation of this link. To avoid link-activation failure because of a mismatch of pre-assigned TG

numbers, the same TG number must be defined by the adjacent node on the adjacent link station (if using pre-assigned TG numbers). If a pre-assigned TG number is defined then the **adj_cp_name** must also be defined (and cannot be set to all zeros) and the **adj_cp_type** must be set to AP_NETWORK_NODE or AP_END_NODE. If zero is entered the TG number is not pre-assigned and is negotiated when the link is activated.

def_data.limited_resource
Specifies whether this link station is to be deactivated when there are no sessions using the link. This is set to one of the following:

AP_NO
The link is not a limited resource and will not be deactivated automatically.

AP_YES or AP_NO_SESSIONS
The link is a limited resource and will be deactivated automatically when no active sessions are using it.

AP_INACTIVITY
The link is a limited resource and will be deactivated automatically when no active sessions are using it, or when no data has flowed on the link for the time period specified by the **link_deact_timer** field. Note that link stations on a nonswitched port cannot be configured as limited resource. A limited resource link station may be configured for CP-CP session support. (This is done by setting this field to AP_YES and **cp_cp_sess_support** to AP_YES.) In this case, if CP-CP sessions are brought up over the link, Personal Communications will not treat the link as a limited resource (and will not bring the link down).

def_data.solicit_sscp_sessions
AP_YES requests the adjacent node to initiate sessions between the SSCP and the local CP and dependent LUs. (In this case the **pu_name** must be set.) AP_NO requests no sessions with the SSCP on this link. This field is only relevant if the link is to an APPN node and is otherwise ignored. If the adjacent node is defined to be a host (**adj_cp_type** is set to AP_HOST_XID3 or AP_HOST_XID0) then Personal Communications always requests the host to initiate sessions between the SSCP and the local CP and dependent LUs (and again the **pu_name** must be set). This field may only be set to AP_YES on a link to an adjacent APPN node if **dspu_services** is set to AP_NONE.

def_data.pu_name
Name of local PU that will use this link if the adjacent node is defined to be a host or **solicit_sscp_sessions** is set to AP_YES on a link to an APPN node. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If the adjacent node is not defined to be a host, and is not defined as an APPN node with **solicit_sscp_sessions** set to AP_YES, this field is ignored.

def_data.disable_remote_act
Specifies whether remote activation of this link is supported (AP_YES or AP_NO).

def_data.dspu_services
Reserved.

def_data.dspu_name
Reserved.

def_data.dlus_name
Reserved.

def_data.bkup_dlus_name
Reserved.

def_data.hpr_supported
Specifies whether HPR is supported on this link (AP_YES or AP_NO). This field is only relevant if the link is to an APPN node and is otherwise ignored.

def_data.hpr_link_lvl_error
Specifies whether HPR traffic should be sent on this link using link-level error recovery (AP_YES or AP_NO). This parameter is ignored if **hpr_supported** is set to AP_NO.

def_data.link_deact_timer
Limited resource link deactivation timer (in seconds). A limited resource link is automatically deactivated if no data traverses the link for the duration of this timer, and no sessions are using the link. If zero is specified for an HPR-capable link then the default value of 30 is used. If zero is specified for a non HPR-capable link then the deactivation timer is not used. Otherwise the minimum value is 5. (If it is set any lower, the specified value will be ignored and 5 will be used.) This parameter is reserved if **limited_resource** is set to AP_NO.

def_data.default_nn_server
Specifies whether a link can be automatically activated by an end node to support CP-CP sessions to a Network Node Server. (AP_YES or AP_NO). Note that the link must be defined to support CP-CP sessions for this field to take effect. This field is ignored if this node is not an End Node.

def_data.ls_attributes
Specifies further information about the adjacent node.

def_data.ls_attributes[0]
Host type.

AP_SNA
Standard SNA host.

AP_FNA
FNA (VTAM-F) host.

AP_HNA
HNA host.

def_data.ls_attributes[1]
Network Name CV suppression option for a link to a back-level LEN node. (This field is ignored unless **adj_cp_type** is set to AP_BACK_LEVEL_LEN_NODE.)

AP_NO
Include Network Name CV in XID3.

AP_SUPPRESS_CP_NAME
Do not include Network Name CV in XID3.

def_data.adj_node_id
Node ID of adjacent node. This a 4-byte hexadecimal string. If **adj_cp_type** indicates the adjacent node is a T2.1 node, this field is ignored unless it is nonzero, and either the **adj_cp_type** is set to AP_BACK_LEVEL_LEN_NODE or the adjacent node does not send a Network Name CV in its XID3. If **adj_cp_type** is set to AP_HOST_XID3 or AP_HOST_XID0, this field is always ignored. If **adj_cp_type** is set to AP_DSPU_XID and this field is nonzero, it is used to check the identity of the downstream PU. If **adj_cp_type** is set to AP_DSPU_NOXID, this field is either ignored (if **dspu_services** is AP_PU_CONCENTRATION) or used to identify the downstream PU to DLUS (if **dspu_services** is AP_DLUR).

def_data.local_node_id
Node ID sent in XIDs on this Link Station. This a 4-byte hexadecimal string. If this field is set to zero, the **node_id** specified on the START_NODE signal will be used in XID exchanges. If this field is nonzero, it will replace the value specified on START_NODE for XID exchanges on this LS.

def_data.cp_cp_sess_support
Specifies whether CP-CP sessions are supported (AP_YES or AP_NO). This field is only relevant if the link is to an APPN node and is otherwise ignored. If **adj_cp_type** is set to AP_BACK_LEVEL_LEN_NODE then it is also ignored and is assumed to be set to AP_NO.

def_data.use_default_tg_chars
Specifies whether the default TG characteristics supplied on the DEFINE_PORT verb should be used (AP_YES or AP_NO). If this is set to AP_YES then the **tg_chars** field will be ignored. This field is only relevant if the link is to an APPN node and is otherwise ignored.

def_data.tg_chars
TG characteristics (See “DEFINE_CN” on page 31). This field is only relevant if the link is to an APPN node and is otherwise ignored.

def_data.target_pacing_count
Numeric value between 1 and 32767, inclusive, indicating the desired pacing window size for BINDs on this TG. The number is only significant when fixed bind pacing is being performed. Personal Communications does not currently use this value.

def_data.max_send_btu_size
Maximum BTU size that can be sent from this link station. This value is used to negotiate the maximum BTU size than can be transmitted between a link station pair. If the link is not HPR-capable then this must be set to a value greater than or equal to 99. If the link is HPR-capable then this must be set to a value greater than or equal to 768.

def_data.ls_role
The link station role that this link station should assume. This may be any one of AP_LS_NEG, AP_LS_PRI or AP_LS_SEC to select a role of negotiable, primary or secondary. The field may also be set to AP_USE_PORT_DEFAULTS to select the value configured on the DEFINE_PORT verb. If the **dlc_type** is AP_TWINAX, then only AP_LS_SEC is supported.

def_data.link_spec_data_len
This field should always be set to zero.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

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

Returned Parameters: State Check

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

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

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_LU_0_TO_3

This verb defines an LU of type 0, 1, 2 or 3. It allows the LU to be added to an LU pool. If the pool does not already exist it is added. This verb cannot be used to modify existing definitions.

VCB Structure

```
typedef struct define_lu_ _to_3
{
        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                       /
        LU_ _TO_3_DEF_DATA def_data;              / defined data                  /
} DEFINE_LU_ _TO_3;

typedef struct lu_ _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    reserv2[8];             / reserved                    /
        unsigned char     app_spec_def_data[16]; / Application Specified Data /
} LU_ _TO_3_DEF_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_LU_0_TO_3

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name Name of the local LU that is being defined. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

def_data.description
Resource description (returned on QUERY_LU_0_TO_3). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_data.nau_address
Network addressable unit address of the LU which must be in the range 1—2554.

def_data.pool_name
Name of LU pool to which this LU belongs. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If the LU does not belong to a pool, this field is set to all binary zeros. If the pool does not currently exist, it is created.

def_data.pu_name
Name of the PU (as specified on the DEFINE_LS verb) that this LU will use. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

def_data.priority
LU priority when sending to the host. This is set to one of the following:

AP_NETWORK
AP_HIGH
AP_MEDIUM
AP_LOW

def_data.lu_model
Model type and number of the LU. This is set to one of the following:

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_UNKNOWN

If a value other than AP_UNKNOWN is specified and the host system supports SDDLU (Self-Defining Dependent LU), the node will generate an unsolicited PSID NMVT reply in order to dynamically define the local LU at the host.

def_data.app_spec_def_data
Application specified defined data. This field is not interpreted by Personal Communications, but is stored and subsequently returned on the QUERY_LU_0_TO_3 verb.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameter:

primary_rc	AP_OK

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_LU_NAME AP_INVALID_NAU_ADDRESS AP_INVALID_PRIORITY

Returned Parameters: State Check

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

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

Returned Parameters: Node Check

If the verb does not execute because the system has not been built with Dependent LU support, Personal Communications returns the following parameter:

primary_rc AP_INVALID_VERB

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_MODE

The DEFINE_MODE verb defines a set of networking characteristics to assign to a particular mode (or group of sessions). This verb can also be used to modify any fields on a previously defined mode. If the SNASVCMG mode is redefined, its **mode_name** and **cos_name** may not be modified. The CPSVCMG mode may not be redefined.

The DEFINE_MODE verb can also be used to define the default COS which unknown modes will be mapped to. This is done by setting **mode_name** to all zeros. The default COS is initially #CONNECT.

Note: It is not necessary to define all the modes you wish to use locally, though they must be defined at your network node and potentially, the partner node. If an ALLOCATE is issued specifying a mode that has not been defined, the node uses the characteristics for the model default mode specified on the DEFINE_DEFAULTS verb. If no such model has been specified, the characteristics of the blank mode are used for the model.

VCB Structure

```
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   reserv1[9];              / reserved                    /
} MODE_CHARS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_MODE

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

mode_name
Name of the mode. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If this is set to all zeros, the default COS is set to **mode_chars.cos_name**, and all other **mode_chars** fields are ignored.

mode_chars.description
Resource description (returned on QUERY_MODE_DEFINITION and QUERY_MODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

mode_chars.max_ru_size_upp
Upper bound for the maximum size of RUs sent and received on sessions in this mode. The value is used when the maximum RU size is negotiated during session activation. The range is 256—16384. This field is ignored if **default_ru_size** is set to AP_YES.

mode_chars.receive_pacing_win
Session pacing window for sessions in this mode. For fixed pacing, this value specifies the receive pacing window. For adaptive pacing, this value is used as an initial receive window size. Note that Personal Communications will always use adaptive pacing unless the adjacent node specifies that it does not support it. The range is 1—63. The value zero is not allowed.

mode_chars.default_ru_size
Specifies whether a default upper bound for the maximum RU size will be used. If this parameter specifies AP_YES, **max_ru_size_upp** is ignored, and the upper bound for the maximum RU size is set to the link BTU size minus the size of the TH and the RH.

AP_YES
AP_NO

mode_chars.max_neg_sess_lim
Maximum number of sessions allowed on this mode between any local LU and partner LU. If a value of zero is specified then there will be no implicit CNOS exchange. The range is 0—32767.

mode_chars.plu_mode_session_limit
Default session limit for this mode. This limits the number of sessions on this mode between any one local LU and partner LU pair. This value is used when CNOS (Change Number of Sessions) exchange is initiated implicitly. If a value of zero is specified then there will be no implicit CNOS exchange. The range is 0—32767.

mode_chars.min_conwin_src
Minimum number of contention winner sessions activatable by any one local LU using this mode. This value is used when CNOS (Change Number of Sessions) exchange is initiated implicitly. If a value of zero is specified then there will be no implicit CNOS exchange. The range is 0—32767.

mode_chars.cos_name
Name of the Class of Service to request when activating sessions on this mode. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

mode_chars.cryptography
Specifies whether session level cryptography must be used (AP_NONE or AP_MANDATORY).

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

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_MIN_CONWINNERS
AP_INVALID_MODE_NAME
AP_INVALID_SESSION_LIMIT
AP_INVALID_RECV_PACING_WINDOW
AP_INVALID_DEFAULT_RU_SIZES
AP_INVALID_SNASVCMG_MODE_LIMIT
AP_MODE_SESS_LIM_EXCEEDS_NEG

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_PARTNER_LU

The DEFINE_PARTNER_LU verb defines the parameters of a partner LU for LU-LU sessions between a local LU and the partner LU. Alternatively, DEFINE_PARTNER_LU can be used to modify all parameters already defined for the partner LU, other than the **fqplu_name** and **plu_alias**.

VCB Structure

```
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   reserv1;                 / reserved                            /
        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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_PARTNER_LU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

plu_chars.fqplu_name
Fully qualified name of the partner LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

plu_chars.plu_alias
Alias of the partner LU. This is an 8-byte string in a locally displayable character set.

plu_chars.description
Resource description (returned on QUERY_PARTNER_LU and QUERY_PARTNER_LU_DEFINITION). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

plu_chars.plu_un_name
Uninterpreted name of the partner LU. This is an 8-byte type-A EBCDIC character string.

plu_chars.max_mc_ll_send_size
Maximum size of LL records sent by and received by mapped conversation services at the partner LU. Range: 1-32767 (32767 is specified by setting this field to 0)

plu_chars.conv_security_ver
Specifies whether the partner LU is authorized to validate **user_ids** on behalf of local LUs, that is whether the partner LU may set the already verified indicator in an Attach request (AP_YES or AP_NO).

plu_chars.parallel_sess_supp
Specifies whether the partner LU supports parallel sessions (AP_YES or AP_NO).

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc	AP_OK

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_DEF_PLU_INVALID_FQ_NAME AP_INVALID_UNINT_PLU_NAME

Returned Parameters: State Check

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

primary_rc	AP_STATE_CHECK
secondary_rc	AP_PLU_ALIAS_CANT_BE_CHANGED AP_PLU_ALIAS_ALREADY_USED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc	AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_PORT

DEFINE_PORT defines a new port or modifies an existing one. This port belongs to a specified DLC which must already have been defined using a DEFINE_DLC verb. The DEFINE_PORT verb provides the port name, which is unique throughout the node, along with port specific parameters and default LS characteristics for use with dynamic link stations. The port specific parameters are concatenated to the basic structure. The default LS characteristics are concatenated immediately following the port specific parameters.

DEFINE_PORT can be used to modify one or more fields on an existing port if the port is in a reset state (after STOP_PORT has been issued) and the **dlc_name** specified on the DEFINE_PORT has not changed since the previous definition of the port.

See “DLC Processes, Ports, and Link Stations” on page 15, for more information about the relationship between DLCs, ports and link stations.

VCB Structure

```
typedef struct define_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                      /
        PORT_DEF_DATA    def_data;                / port defined data                 /
} DEFINE_PORT;

typedef struct port_def_data
{
        unsigned  char   description[RD_LEN]
                                                  / resource description              /
        unsigned char    dlc_name[8];             / DLC name associated with port /
        unsigned char    port_type;               / port type                         /
        unsigned  char   reserv3[7];              / 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 lim    /
        unsigned char    ls_role;                 / initial link station role        /
        unsigned  char   reserv1[27];             / reserved                          /
        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 lim        /
        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_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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_PORT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

port_name
Name of port being defined. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set.

def_data.description
Resource description (returned on QUERY_PORT). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_data.dlc_name
Name of associated DLC which is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. This named DLC must have already been defined by a DEFINE_DLC verb.

def_data.port_type
Specifies the type of line used by the port. The value corresponds to one of the following:

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

Note that if this field is set to AP_PORT_SATF then the **ls_role** must be set to AP_LS_NEG.

def_data.port_number
Port number.

def_data.max_rcv_btu_size
Maximum BTU size that can be received. If implicit HPR-capable links are not supported on the port then this must be set to a value greater than or equal to 99. If implicit HPR-capable links are supported on the port then this must be set to a value greater than or equal to 768.

def_data.tot_link_act_lim
Total link activation limit. This specifies the maximum number of link stations that can be active concurrently. This must be greater than or equal to the sum of the **inb_link_act_lim** and **out_link_act_lim** fields. If the **port_type** is set to AP_PORT_NONSWITCHED and the **ls_role** is set to AP_LS_NEG or AP_LS_SEC then this field must be set to one. If the **ls_role** is set to AP_LS_PRI then this field must be in the range greater than or equal to one to 256.

def_data.inb_link_act_lim
Inbound link activation limit. This specifies the number of link stations reserved for inbound activation on this port. The maximum number of outbound link stations that can be active concurrently is therefore **def_data.tot_link_act_lim** - **def_data.inb_link_act_lim**. If the **port_type** is set to AP_PORT_NONSWITCHED and the **ls_role** is set to AP_LS_NEG or AP_LS_PRI then this field must be set to zero. If the **port_type** is set to AP_PORT_NONSWITCHED and the **ls_role** is set to AP_LS_SEC then this field must be set to zero or one.

def_data.out_link_act_lim
Outbound link activation limit. This specifies the number of link stations reserved for outbound activation on this port. The maximum number of inbound link stations that can be active concurrently is therefore **def_data.tot_link_act_lim** - **def_data.out_link_act_lim**. If the **port_type** is set to AP_PORT_NONSWITCHED and the **ls_role** is set to AP_LS_NEG then this field must be set to zero. If the **ls_role** is set to AP_LS_PRI then this field must be equal to **tot_link_act_lim**. If the **port_type** is set to AP_PORT_NONSWITCHED and the **ls_role** is set to AP_LS_SEC then this field must be set to zero or one.

def_data.ls_role
Link station role. This can be negotiable (AP_LS_NEG), primary (AP_LS_PRI), or secondary (AP_LS_SEC). The link station role determines the relationship between the values specified by the **tot_act_lim**, **inb_link_act_lim**, and **out_link_act_lim** fields as described above. Note that if the **port_type** is set to AP_PORT_SATF then the **ls_role** must be set to AP_LS_NEG.

def_data.implicit_deact_timer
Limited resource link deactivation timer (in seconds). A limited resource HPR-capable implicit link is automatically deactivated if no data traverses the link for the duration of this timer, and no sessions are using the link. If zero is specified the default value of 30 is used. Otherwise the minimum value is 5. (If it is set any lower, the specified value will be ignored and 5 will be used.) Note that this parameter is reserved unless **implicit_limited_resource** and **implicit_hpr_support** are both set to AP_YES.

def_data.act_xid_exchange_limit
Activation XID exchange limit.

def_data.nonact_xid_exchange_limit
Non-activation XID exchange limit.

def_data.ls_xmit_rcv_cap
Specifies the link station transmit/receive capability. This is either two-way simultaneous (AP_LS_TWS) or two way alternating (AP_LS_TWA).

def_data.max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgement is sent. The range is 1—127.

def_data.target_pacing_count
Numeric value between 1 and 32767 inclusive indicating the desired pacing window size for BINDs on this TG. The number is only significant when fixed bind pacing is being performed. Note that Personal Communications does not currently use this value.

def_data.max_send_btu_size
Maximum BTU size that can be sent from this link station. This value is used to negotiate the maximum BTU size than can be transmitted between a link station pair. If implicit HPR-capable links are not supported on the port then this must be set to a value greater than or equal to 99. If implicit HPR-capable links are supported on the port then this must be set to a value greater than or equal to 768.

def_data.dlc_data.length
Port address length.

def_data.dlc_data.address
Port address.

def_data.hpr_dlc_data.length
HPR Port address length.

def_data.hpr_dlc_data.address
HPR Port address. This is currently used when supporting HPR links. The field specifies the information sent by Personal Communications in the X'80' subfield of the X'61' control vector on XID3s exchanged on link stations using this port.

def_data.implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are permitted for implicit link stations off this port (AP_YES or AP_NO).

def_data.implicit_limited_resource
Specifies whether implicit link stations off this port should be defined as limited resources (AP_YES or AP_NO).

def_data.implicit_hpr_support
Specifies whether HPR should be supported on implicit links (AP_YES or AP_NO).

def_data.implicit_link_lvl_error
Specifies whether HPR traffic should be sent on implicit links using link-level error recovery (AP_YES or AP_NO). Note that the parameter is reserved if **implicit_hpr_support** is set to AP_NO.

def_data.default_tg_chars
TG characteristics (See “DEFINE_COS” on page 35). These are used for implicit link stations off this port and also for defined link stations which specify **use_default_tg_chars**.

def_data.discovery_supported
Specifies whether Discovery search functions are to be performed on this port (AP_YES or AP_NO).

def_data.port_spec_data_len
Length of data to be passed unchanged to port on ACTIVATE_PORT signal. The data should be concatenated to the basic structure.

def_data.link_spec_data_len
This field should always be set to zero.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

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_HPR_NOT_SUPPORTED

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc
AP_PORT_ACTIVE
AP_DUPLICATE_PORT_NUMBER

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_TP

The DEFINE_TP verb defines transaction program (TP) information for use by the Node Operator Facility TP Attach Manager when it processes incoming attaches from partner LUs. This verb can also be used to modify one or more fields on a previously defined TP (but cannot be used to modify Personal Communications defined TPs).

VCB Structure

```
typedef struct define_tp
{
        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    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   reserv3[1 ];         / 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 reserved[6];            / max size is 12  bytes            /
} TP_SPEC_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_TP

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

tp_name Name of the TP being defined. Note that Personal Communications does not check the character set of this field.

tp_chars.description
Resource description (returned on QUERY_TP_DEFINITION and QUERY_TP). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

tp_chars.conv_type
Specifies the type(s) of conversation supported by this TP.

AP_BASIC
AP_MAPPED
AP_EITHER

tp_chars.security_rqd
Specifies whether conversation security information is required to start the TP (AP_NONE, AP_SAME or AP_PGM).

tp_chars.sync_level
Specifies the synchronization levels supported by the TP.

AP_NONE
The TP supports a synchronization level of None.

AP_CONFIRM_SYNC_LEVEL
The TP supports a synchronization level of Confirm.

AP_EITHER
The TP supports a synchronization level of None or Confirm.

tp_chars.dynamic_load
Specifies whether the TP can be dynamically loaded (AP_YES or AP_NO).

tp_chars.enabled
Specifies whether the TP can be attached successfully (AP_YES or AP_NO).

tp_chars.pip_allowed
Specifies whether the TP can receive program initialization (PIP) parameters (AP_YES or AP_NO).

tp_chars.tp_instance_limit
Limit on the number of concurrently active TP instances. A value of zero means no limit.

tp_chars.incoming_alloc_timeout
Specifies the number of seconds that an incoming attach will be queued waiting for a RECEIVE_ALLOCATE. Zero implies no timeout, and so it will be held indefinitely.

tp_chars.rcv_alloc_timeout
Specifies the number of seconds that a RECEIVE_ALLOCATE verb will be queued while waiting for an Attach. Zero implies no timeout, and so it will be held indefinitely.

tp_chars.tp_data_len
Length of the implementation dependent TP data.

tp_spec_data
Information used by the Attach Manager when launching the Transaction program. See the Attach Manager in **Personal Communications Client/Server Communications Programming** for further details of how this is used.

tp_chars.tp_data.pathname
Specifies the path and TP name.

tp_chars.tp_data.parameters
Specifies the parameters for the TP.

tp_chars.tp_data.queued
Specifies whether the TP will be queued.

tp_chars.tp_data.load_type
Specifies how the TP will be loaded.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_ADJACENT_LEN_NODE

DELETE_ADJACENT_LEN_NODE removes entries in the node directory database which are associated with the resources on an adjacent LEN Node.

To remove the LEN Node's CP from the directory along with its LUs, set **num_of_lus** to zero. If **num_of_lus** is nonzero, this verb is used to remove LEN Node LUs from the directory, leaving the CP definition intact. Up to ten LUs may be deleted at once. The LUs specified on a single verb must either be all wildcard entries, or all explicit entries.

If the verb fails for any reason, no directory entries will be deleted.

VCB Structure

```
typedef struct delete_adjacent_len_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     num_of_lus;      /  number of LUs            /
        unsigned char     lu_names[1 ][8]; /  LU names                /
        unsigned char     wildcard_lus;    /  wildcard LUs             /
} DELETE_ADJACENT_LEN_NODE;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_ADJACENT_LEN_NODE

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

cp_name The fully qualified name of the CP in the adjacent LEN end node. The name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

num_of_lus The number of LUs to be deleted, in the range 1 to 10. Set this to zero if the entire LEN Node definition is to be deleted.

lu_names The names of the LUs on the LEN Node to be deleted. Each name is an 8-byte type-A EBCDIC character string and is right-padded with EBCDIC spaces. Valid only if **num_of_lus** is nonzero.

wildcard_lus Indicates whether the LU names to be deleted are wildcards.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_CP_NAME
AP_INVALID_LU_NAME
AP_INVALID_NUM_LUS

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_INVALID_CP_NAME
AP_INVALID_LU_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_CN

DELETE_CN deletes and frees the memory for a connection network control block if all the associated ports are reset. DELETE_CN can also be used to delete selected ports from a connection network. To do this, the user should set the **num_ports** field to a nonzero value and supply the port names of the ports to be deleted.

VCB Structure

```
typedef struct delete_cn
{
        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    fqcn_name[17];   / name of Connection Network   /
        unsigned char    reserv1;         / reserved                     /
        unsigned short   num_ports;       / number of ports to delete    /
        unsigned char    port_name[MAX_PORTS_PER_DEFINE] [8];
                                          / names of ports to delete     /
} DELETE_CN;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_CN

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

fqcn_name Name of connection network (17 bytes long) to be deleted. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

num_ports The number of ports to delete on the connection network. This should be set to zero if the entire connection network is to be deleted.

port_name Names of the ports to be deleted if the **num_ports** is nonzero. Each port name is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If the **num_ports** field is zero this field is reserved.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_CN_NAME AP_INVALID_NUM_PORTS_SPECIFIED

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc	AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

DELETE_COS

DELETE_COS deletes a Class of Service entry unless it is one of the default Classes of Service defined by SNA.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_COS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

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

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_COS_NAME_NOT_DEFD
AP_SNA_DEFD_COS_CANT_BE_DELETE

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_DLC

DELETE_DLC deletes all ports, link stations and connection network transmission groups (TGs) associated with the DLC if it is reset. All DLC control blocks are deleted and the memory freed. The Node Operator Facility returns a response specifying whether the DLC was deleted successfully.

Note that if a link station, which has a PU associated with it, is deleted (because it is associated with the DLC) then any LUs defined on this PU will also be deleted.

VCB Structure

```
typedef struct delete_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            /

} DELETE_DLC;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_DLC

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

dlc_name Name of DLC to be deleted. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_DLC_NAME

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_DLC_ACTIVE

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_FOCAL_POINT

The DELETE_FOCAL_POINT verb can be used to delete focal points of a specified type and category. For more information about focal point types, see "DEFINE_FOCAL_POINT" on page 50. If an active focal point is deleted it will be revoked. To revoke the active focal point (of any type) specify a type of AP_ACTIVE. If a backup or implicit focal point is deleted (by specifying AP_BACKUP or AP_IMPLICIT) when it is not currently active, any information stored about it will simply be removed.

Note that the DEFINE_FOCAL_POINT verb may also be used to revoke currently active focal points. This duplicated function is retained for back compatibility.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_FOCAL_POINT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

ms_category Management Services category. This may either be one of the four-byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services categories as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation defined name.

type Specifies the type of the focal point that is being deleted. Possible types are:

AP_ACTIVE
The currently active focal point (which may be of any type) is revoked.

AP_IMPLICIT
The implicit definition is removed. If the currently active focal point is an implicit focal point, then it is revoked.

AP_BACKUP
The backup definition is removed. If the currently active focal point is a backup focal point, then it is revoked.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_TYPE
AP_INVALID_CATEGORY_NAME

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_INTERNAL_PU

The DELETE_INTERNAL_PU verb requests the deletion of a DLUR-served local PU. The verb will only succeed if the PU does not have an active SSCP-PU session.

Any LUs associated with the PU will be deleted.

VCB Structure

```
typedef struct delete_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       /
} DELETE_INTERNAL_PU;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_INTERNAL_PU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

pu_name Name of the internal PU that is being deleted. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_PU_NAME
AP_INVALID_PU_TYPE

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_PU_NOT_RESET

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LOCAL_LU

The DELETE_LOCAL_LU verb requests deletion of the local LU definition.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_LOCAL_LU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name Name of the local LU that is being defined. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_LU_NAME
AP_CANT_DELETE_CP_LU

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LS

DELETE_LS checks that the link station has been previously defined and reset. It removes the link station control block and returns a response from the Node Operator Facility specifying whether the link station has been deleted successfully. Note that any LUs defined on the PU using this link station will also be deleted.

VCB Structure

```
typedef struct delete_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   /
} DELETE_LS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_LS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

ls_name Name of link station being deleted. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_LINK_NAME

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_LS_ACTIVE
AP_INVALID_LINK_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LU_0_TO_3

This verb is used to delete a specific LU.

VCB Structure

```
typedef struct delete_lu_ _to_3
{
        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                 /
} DELETE_LU_ _TO_3;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_LU_0_TO_3

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name Name of the LU to be deleted. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_LU_NAME

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_INVALID_LU_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_MODE

The DELETE_MODE verb requests deletion of a mode definition. Default definitions for CPSVCMG, SNASVCMG, and other standard SNA modes will not be deleted.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_MODE

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

mode_name Name of the mode. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

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

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_PARTNER_LU

The DELETE_PARTNER_LU requests the deletion of a partner LU definition.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_PARTNER_LU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

fqplu_name Fully qualified name of the partner LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_PLU_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_PORT

DELETE_PORT deletes all link stations and connection network transmission groups (TGs) associated with the port if it is reset. It then deletes the port's control block, frees the memory, and returns a response from the Node Operator Facility indicating whether the port has been deleted successfully.

Note that if a link station, which has a PU associated with it, is deleted (because it is associated with the port) then any LUs defined on this PU will also be deleted.

VCB Structure

```
typedef struct delete_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            /
} DELETE_PORT;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_PORT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

port_name Name of port being deleted. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_PORT_NAME

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_PORT_ACTIVE

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_TP

The DELETE_TP requests the deletion of a transaction program (TP) definition.

VCB Structure

```
typedef struct delete_tp
{
        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    tp_name[64];     / TP name                /
} DELETE_TP;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_TP format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

tp_name Name of the transaction program. Personal Communications does not check the character set of this field.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_TP_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_TP

Chapter 5. Activation and Deactivation Verbs

This chapter describes verbs that are used to activate and deactivate:

- data link controls (DLCs)
- ports
- link stations
- sessions
- conversation groups

This chapter also describes a verb used to request a path switch to a connection that supports High-Performance Routing (HPR).

© Copyright IBM Corp. 1989, 1996

START_DLC

START_DLC requests the activation of a data link control (DLC). It is subsequently returned indicating whether the activation of the DLC was successful. Note that the DLC can be started even if no ports have been defined for it. See “DLC Processes, Ports, and Link Stations” on page 15, for more information about the relationship between DLCs, ports and link stations.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_START_DLC

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

dlc_name Name of Data Link Control instance that is to be started. This is an 8- byte string in a locally displayable character set which must have already been defined by a DEFINE_DLC verb.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_DLC

Returned Parameters: State Check

If the verb does not execute because the DLC is deactivating , Personal Communications returns the following parameters:

primary_rc AP_STATE_CHECK

secondary_rc AP_DLC_DEACTIVATING

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

START_INTERNAL_PU

The START_INTERNAL_PU verb requests the dependent LU requester (DLUR) to initiate SSCP-PU session activation for a previously defined local PU which is served by DLUR.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_START_INTERNAL_PU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

pu_name Name of the internal PU for which the SSCP-PU session activation flows will be solicited. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

dlus_name Name of the dependent LU server (DLUS) node that DLUR will contact to solicit SSCP-PU session activation for the given PU. This should be set to all zeros or a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the field is set to all zeros, the DLUS specified in the DEFINE_INTERNAL_PU verb will be used. If no DLUS has been specified in the DEFINE_INTERNAL_PU verb, then the global default (if specified by a DEFINE_DLUR_DEFAULTS verb) will be used.

bkup_dlus_name Name of DLUS node that DLUR will store as the backup DLUS for the given PU. This should be set to all zeros or a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the field is set to all zeros, the backup DLUS name specified by a DEFINE_INTERNAL_PU verb will be retained as the backup DLUS for this PU. If no backup DLUS was specified by the DEFINE_INTERNAL_PU verb, the global backup default

DLUS (if defined by the DEFINE_DLUR_DEFAULTS verb) is retained as the backup default for this PU.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc	AP_OK

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_DLUS_NAME AP_INVALID_BKUP_DLUS_NAME

Returned Parameters: State Check

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

primary_rc	AP_STATE_CHECK
secondary_rc	AP_NO_DEFAULT_DLUS_DEFINED AP_PU_NOT_DEFINED AP_PU_ALREADY_ACTIVATING AP_PU_ALREADY_ACTIVE

Returned Parameters: Other Conditions

If the verb does not execute successfully, Personal Communications returns the following parameters:

primary_rc	AP_UNSUCCESSFUL
secondary_rc	AP_DLUS_REJECTED AP_DLUS_CAPS_MISMATCH AP_PU_FAILED_ACTPU

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

START_LS

START_LS requests activation of a link. It is returned as a response specifying whether the link was successfully activated.

See “DLC Processes, Ports, and Link Stations” on page 15, for more information about the relationship between DLCs, ports and link stations.

VCB Structure

```
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[34];     / reserved                     /
} START_LS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_START_LS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

ls_name Name of link station to be started. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. The value of **ls_name** must match that on the DEFINE_LS verb.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_LINK_NAME_SPECIFIED

Returned Parameters: State Check

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

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

Returned Parameters: Cancelled

If the verb does not execute because it was cancelled by a subsequent STOP_LS or STOP_PORT before the link became active, Personal Communications returns the following parameters

primary_rc AP_CANCELLED

secondary_rc AP_LINK_DEACTIVATED

Returned Parameters: Link Failed to Start

If the verb does not execute because the partner could not be found by the link software, Personal Communications returns the following parameters:

primary_rc AP_LS_FAILURE

secondary_rc AP_PARTNER_NOT_FOUND

If the verb does not execute because a link error occurred while the link was being established, Personal Communications returns the following parameters:

primary_rc AP_LS_FAILURE

secondary_rc AP_ERROR

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

START_PORT

START_PORT requests the activation of a port. It is returned indicating whether the port was successfully activated. The port can be started even if no link stations have been defined for it, but it will not be started if its parent DLC is inactive.

See “DLC Processes, Ports, and Link Stations” on page 15, for more information about the relationship between DLCs, ports and link stations.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_START_PORT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

port_name Name of port to be started. This is an 8-byte string in a locally displayable character set and must match that on the DEFINE_PORT verb.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_PORT_NAME

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_DLC_INACTIVE
AP_STOP_PORT_PENDING
AP_DUPLICATE_PORT

Returned Parameters: Cancelled

If the verb does not execute because it was cancelled, Personal Communications returns the following parameters:

primary_rc AP_CANCELLED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

STOP_DLC

STOP_DLC requests that a DLC be stopped. It is returned indicating whether the DLC was successfully stopped.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_STOP_DLC

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

stop_type Manner in which DLC should be stopped.

AP_ORDERLY_STOP
Node should perform cleanup operations before stopping DLC.

AP_IMMEDIATE_STOP
Node should stop DLC immediately.

dlc_name Name of DLC to be stopped. This is an 8-byte string in a locally displayable character set which must match that on the DEFINE_DLC verb.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_DLC
AP_UNRECOGNIZED_DEACT_TYPE

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_STOP_DLC_PENDING

Returned Parameters: Cancelled

If the verb does not execute because it has been cancelled, Personal Communications returns the following parameters:

primary_rc AP_CANCELLED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

STOP_INTERNAL_PU

The STOP_INTERNAL_PU verb requests the dependent LU requester (DLUR) initiate SSCP-PU session deactivation for a previously defined local PU which is served by DLUR.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_STOP_INTERNAL_PU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

pu_name Name of the internal PU for which the SSCP-PU session will be deactivated. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

stop_type Specifies stop type requested for the PU. An orderly stop will deactivate all underlying PLU-SLU and SSCP-LU sessions before deactivating the SSCP-PU session.

AP_ORDERLY_STOP
AP_IMMEDIATE_STOP

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_STOP_TYPE

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_PU_NOT_DEFINED
AP_PU_ALREADY_DEACTIVATING
AP_PU_NOT_ACTIVE

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

STOP_LS

STOP_LS requests the deactivation of a link station. It is returned specifying whether the link was stopped successfully.

VCB Structure

```
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    resrved[3];      / reserved                         /
} STOP_LS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_STOP_LS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

stop_type Manner in which the link station should be stopped.

AP_ORDERLY_STOP
Node should perform cleanup operations before stopping the link station.

AP_IMMEDIATE_STOP
Node should stop the link station immediately.

ls_name Name of link station to be stopped. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. The value of **ls_name** must match that on the DEFINE_LS verb.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_UNRECOGNIZED_DEACT_TYPE
AP_LINK_NOT_DEFD

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_LINK_DEACT_IN_PROGRESS

Returned Parameters: Cancelled

If the verb does not execute because it was cancelled, Personal Communications returns the following parameters:

primary_rc AP_CANCELLED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

STOP_PORT

STOP_PORT requests that a port be stopped. It is returned specifying whether the port was stopped successfully.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_STOP_PORT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

stop_type Manner in which the port should be stopped.

AP_ORDERLY_STOP
Node should perform cleanup operations before stopping the port.

AP_IMMEDIATE_STOP
Node should stop the port immediately.

port_name Name of port to be stopped. This is an 8-byte string in a locally displayable character set which must match that on the DEFINE_PORT verb.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_PORT_NAME
AP_UNRECOGNIZED_DEACT_TYPE

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_STOP_PORT_PENDING

Returned Parameters: Cancelled

If the verb does not execute because it has been cancelled, Personal Communications returns the following parameters:

primary_rc AP_CANCELLED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

ACTIVATE_SESSION

The ACTIVATE_SESSION verb requests activation of a session between the local LU and a specified partner LU using the characteristic of a particular mode.

VCB Structure

```
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   reserv3;         / reserved                 /
        unsigned char   session_id[8];   / session identifier       /
} ACTIVATE_SESSION;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_ACTIVATE_SESSION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name LU name of the local LU requested to activate a session. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias Alias of the local LU requested to activate a session. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_alias** and the **lu_name** are set to all zeros then the verb is forwarded to the LU associated with the CP (the default LU).

plu_alias Alias by which the partner LU is known to the local LU. This name must match the name of a partner LU established during configuration. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If this field is set to all zeros, the **fqplu_name** field is used to specify the required partner LU.

mode_name Name of a set of networking characteristics defined during configuration. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

fqplu_name Fully qualified LU name for the partner LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is only significant if the **plu_alias** field is set to all zeros.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

secondary_rc AP_AS_SPECIFIED
AP_AS_NEGOTIATED

session_id 8-byte identifier of the activated session.

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_PLU_NAME
AP_LU_SESS_LIMIT_EXCEEDED
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_MODE_NAME

If the verb exceeds the session limit for the mode, Personal Communications returns the following parameters:

primary_rc AP_PARAMETER_CHECK

Secondary_rc AP_EXCEEDS_MAX_ALLOWED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

Returned Parameters: Other Errors

If the verb does not execute because of other errors, Personal Communications returns one of the following parameters:

primary_rc AP_ACTIVATION_FAIL_NO_RETRY
AP_ACTIVATION_FAIL_RETRY

DEACTIVATE_CONV_GROUP

The DEACTIVATE_CONV_GROUP verb requests the deactivation of the session corresponding to the specified conversation group. Although this verb is part of the Node Operator Facility API, it is primarily intended for use by application programmers writing TPs which utilize the Personal Communications APPC API. The conversation group identifier is returned by the MC_ALLOCATE, ALLOCATE, MC_GET_ATTRIBUTES, GET_ATTRIBUTES and RECEIVE_ALLOCATE verbs defined in *Personal Communications Client Server Communications*.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEACTIVATE_CONV_GROUP

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name LU name of the local LU requested to deactivate the conversation group. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias Alias of the local LU requested to deactivate the conversation group. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_name** and **lu_alias** are set to all zeros, the verb is forwarded to the LU associated with the CP (the default LU).

conv_group_id Conversation group identifier for the session to be deactivated.

type Type of deactivation.

AP_DEACT_CLEANUP
The session is terminated immediately, without waiting for a response from the partner LU.

AP_DEACT_NORMAL
The session terminates after all conversations using the session are ended.

sense_data Specifies the sense data for use in the CLEANUP type of deactivation.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_CLEANUP_TYPE
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEACTIVATE_SESSION

The DEACTIVATE_SESSION verb requests the deactivation of a particular session, or all sessions on a particular mode.

VCB Structure

```
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[2 ];     / reserved                   /
} DEACTIVATE_SESSION;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEACTIVATE_SESSION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name LU name of the local LU requested to deactivate a session. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias Alias of the local LU requested to deactivate a session. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_name** and the **lu_alias** fields are set to all zeros then the verb is forwarded to the LU associated with the CP (the default LU).

session_id 8-byte identifier of the session to deactivate. If this field is set to all zeros, Personal Communications deactivates all sessions for the partner LU and mode.

plu_alias Alias by which the partner LU is known to the local LU. This name must match the name of a partner LU established during configuration. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If this field is set to all zeros, the **fqplu_name** field is used to specify the required partner LU.

mode_name Name of a set of networking characteristics defined during configuration. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

type Type of deactivation.

AP_DEACT_CLEANUP
Terminates the session immediately, without waiting for a response from the partner LU.

AP_DEACT_NORMAL
Terminates the session after all conversations using the session are ended.

sense_data Specifies the sense data to be used for the CLEANUP type of deactivation.

fqplu_name Fully qualified LU name for the partner LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is only significant if the **plu_alias** field is set to all zeros.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameter:

primary_rc AP_OK

Note that if the **session_id** cannot be matched with any existing sessions, it is assumed that this is because the session has already been deactivated. In this case the verb completes successfully.

Returned Parameters: Parameter Check

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

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

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

PATH_SWITCH

The PATH_SWITCH verb requests Personal Communications to path switch routes on a connection which supports high-performance routing (HPR). If a better path cannot be found, the connection is left unchanged.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_PATH_SWITCH

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

rtp_connection_name Identifies the RTP connection to path-switch. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameter:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_RTP_CONNECTION

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_PATH_SWITCH_IN_PROGRESS

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

Chapter 6. Query Verbs

This chapter describes verbs used to query information about node configuration and status.

© Copyright IBM Corp. 1989, 1996

QUERY_CN

QUERY_CN returns information about adjacent Connection Networks. This information is structured as “determined data” (data gathered dynamically during execution) and “defined data” (the data supplied by the application on DEFINE_CN).

The information is returned as a formatted list. To obtain information about a specific CN, or to obtain the list information in several “chunks,” the **fqcn_name** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

This list is ordered on the **fqcn_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering).

If AP_LIST_FROM_NEXT is selected, the list will start from the next entry according to the defined ordering (whether the specified entry exists or not).

VCB Structure

```
typedef struct query_cn
{
        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 /
} 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[2 ];             / 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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_CN

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information: The **fqcn_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

fqcn_name Fully qualified, 17-byte, connection network name. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

cn_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

cn_data.fqcn_name
Fully qualified, 17-byte, connection network name. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

cn_data.det_data.num_act_ports
Dynamic value giving number of active ports on the connection network.

cn_data.def_data.description
Resource description (as specified on DEFINE_CN). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

cn_data.def_data.num_ports
Number of ports on the connection network.

cn_data.def_data.tg_chars.effect_cap
Actual units of effective capacity. The value is encoded as a 1-byte floating point number, represented by the formula 0.1mmm * 2 eeeee where the bit representation of the byte is eeeeemmm. Each unit of effective capacity is equal to 300 bits per second.

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

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

cn_data.def_data.tg_chars.security
Security values as described in the list below.

AP_SEC_NONSECURE
No security exists.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data transmitted over this connection network will flow over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

AP_SEC_SECURE_CONDUIT
The line is a secure conduit that is not guarded.

AP_SEC_GUARDED_CONDUIT
Conduit is protected against physical tapping.

AP_SEC_ENCRYPTED
Encryption over the line.

AP_SEC_GUARDED_RADIATION
Line is protected against physical and radiation tapping.

cn_data.def_data.tg_chars.prop_delay
Propagation delay representing the time it takes for a signal to travel the length of the link, in microseconds. The value is encoded as a 1-byte floating point number, represented by the formula 0.1mmm * 2 eeeee where the bit representation of the byte is eeeeemmm. Default values are listed below.

AP_PROP_DELAY_MINIMUM
No propagation delay.

AP_PROP_DELAY_LAN
Less than 480 microseconds delay.

AP_PROP_DELAY_TELEPHONE
Between 480 and 49 512 microseconds delay.

AP_PROP_DELAY_PKT_SWITCHED_NET
Between 49 512 and 245 760 microseconds delay.

AP_PROP_DELAY_SATELLITE
Longer than 245 760 microseconds delay.

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

cn_data.def_data.tg_chars.modem_class
Reserved. This field should always be set to zero.

cn_data.def_data.tg_chars.user_def_parm_1
User defined parameter in range 0—255.

cn_data.def_data.tg_chars.user_def_parm_2
User defined parameter in range 0—255.

cn_data.def_data.tg_chars.user_def_parm_3
User defined parameter in range 0—255.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_CN_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_CN_PORT

QUERY_CN_PORT returns information about ports defined on adjacent Connection Networks. The information is returned as a formatted list. To obtain information about a specific port, or to obtain the list information in several "chunks," the **port_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. Note that the **fqcn_name** field must always be set to the name of a valid connection network.

See "Querying the Node" on page 12, for background on how the list formats are used.

VCB Structure

```
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[2 ];         / reserved                     /
} CN_PORT_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_CN_PORT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information: The combination of **fqcn_name** and **port_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

fqcn_name Fully qualified, 17-byte, connection network name. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field must always be set.

port_name 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

cn_port_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

cn_port_data.fqcn_name
Fully qualified, 17-byte, connection network name. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

cn_port_data.port_name
Port name in an 8-byte, locally displayable character set. All 8 bytes are significant.

cn_port_data.tg_num
Transmission group number for specified port.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_CN_NAME AP_INVALID_PORT_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_COS

QUERY_COS returns route calculation information for a specific Class of Service. The information is returned as a formatted list. To obtain information about a specific COS, or to obtain the list information in several “chunks,” the **cos_name** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used. This list is ordered on the **cos_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with IBM's 6611 APPN MIB ordering). If AP_LIST_FROM_NEXT is selected the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

VCB Structure

```
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  char   reserv2[2];              / reserved                            /
        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[2 ];             / reserved                            /
} COS_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_COS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information: The **cos_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

cos_name Class of Service name. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

cos_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

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

cos_data.description
Resource description (as specified on DEFINE_COS). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

cos_data.transmission_priority
Transmission priority. This is set to one of the following:

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

cos_data.num_of_node_rows
Number of node rows for this COS.

cos_data.num_of_tg_rows
Number of TG rows for this COS.

cos_data.trees
Number of route tree caches built for this COS since the last initialization.

cos_data.calcs
Number of session activation requests (and therefore route calculations) specifying this Class of Service.

cos_data.rejs
Number of session activation requests that failed because there was no acceptable (using the specified Class of Service) route from this node to the named destination through the network. A route is only acceptable if it is made up entirely of active TGs and nodes that can provide the specified Class of Service.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_COS_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DEFAULT_PU

QUERY_DEFAULT_PU allows the user to query the default PU defined using a DEFINE_DEFAULT_PU verb.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_DEFAULT_PU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

def_pu_name Name of the PU specified on the most recent DEFINE_DEFAULT_PU verb. This is an 8-byte alphanumeric type A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If no DEFINE_DEFAULT_PU verb has been issued then this field will be set to all zeros.

description Resource description (as specified on DEFINE_DEFAULT_PU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_pu_sess Name of the PU associated with the currently active default PU session. This will be different from the **def_pu_name** field if a default PU has been defined, but the session associated with it is not active. In this case, Personal Communications continues to use the session associated with the previous default PU until the session associated with the defined default PU becomes active. If there are no active PU sessions then this field will be set to all zeros.

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DEFAULTS

QUERY_DEFAULTS allows the user to query the defaults defined using the DEFINE_DEFAULTS verb.

VCB Structure

```
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    reserv[248];     / reserved                 /
} DEFAULT_CHARS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_DEFAULTS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

default_chars.description
Resource description (as specified on DEFINE_DEFAULTS). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

default_chars.mode_name
Name of the mode specified on the most recent DEFINE_DEFAULTS verb. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If no DEFINE_DEFAULTS_PU verb has been issued then this field will be set to all zeros.

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DIRECTORY_LU

QUERY_DIRECTORY_LU returns a list of LUs from the directory database. The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific LU, or to obtain the list information in several "chunks," the **lu_name** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered by the **lu_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with IBM's 6611 APPN MIB ordering). If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

VCB Structure

```
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 qualif ied 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    reserva[2 ];             / reserved /
} DIRECTORY_LU_DETAIL;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_DIRECTORY_LU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB, in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **lu_name** specified (see below) represents an index value that is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

lu_name Network qualified LU name. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of directory entries returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

directory_lu_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

directory_lu_summary.lu_name
Network qualified LU name. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

directory_lu_summary.description
Resource description (as specified on DEFINE_LOCAL_LU, or DEFINE_ADJACENT_LEN_NODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

directory_lu_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

directory_lu_detail.lu_name
Network qualified LU name. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

directory_lu_detail.description
Resource description (as specified on DEFINE_LOCAL_LU, or DEFINE_ADJACENT_LEN_NODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

directory_lu_detail.server_name
Network qualified name of the node serving the LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

directory_lu_detail.lu_owner_name
Network qualified name of the node owning the LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

directory_lu_detail.location
Specifies the location of the resource which may be one of the following:

AP_LOCAL
The resource is at the local node.

AP_DOMAIN
The resource belongs to an attached end node.

AP_CROSS_DOMAIN
The resource is not within the domain of the local node.

directory_lu_detail.entry_type
Specifies the type of the directory entry. This can be one of the following:

AP_HOME
Local resource.

AP_CACHE
Cached entry.

AP_REGISTER
Registered resource (NN only).

directory_lu_detail.wild_card
Specifies the type of wildcard the LU will match.

AP_OTHER
Unknown type of LU entry.

AP_EXPLICIT
The full **lu_name** will be used for locating this LU.

AP_PARTIAL_WILDCARD
Only the nonspace portions of **lu_name** will be used for locating this LU.

AP_FULL_WILDCARD
All **lu_names** will be directed to this LU.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_LU_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DIRECTORY_STATS

QUERY_DIRECTORY_STATS returns directory database statistics. (The statistics that refer to cache information are reserved in the case of an End Node). The verb can be used to gauge the level of network locate traffic. In the case of a Network Node this information can be used to tune the size of the directory cache which is configurable at node initialization time.

VCB Structure

```
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[2 ];         / reserved                         /
} QUERY_DIRECTORY_STATS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_DIRECTORY_STATS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

max_caches Reserved.

cur_caches Reserved.

cur_home_entries
Current number of home entries.

cur_reg_entries
Current number of registered entries.

cur_directory_entries
Total number of entries currently in the directory.

cache_hits Reserved.

cache_misses
Reserved.

in_locates Number of directed locates received.

in_bcast_locates
Number of broadcast locates received.

out_locates Number of directed locates sent.

out_bcast_locates
Number of broadcast locates sent.

not_found_locates
Number of directed locates returned with a 'not found'.

not_found_bcast_locates
Number of broadcast locates returned with a 'not found'.

locates_outstanding
Current number of outstanding locates, both directed and broadcast.

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLC

QUERY_DLC returns a list of information about the DLCs defined at the node. This information is structured as “determined data” (data gathered dynamically during execution) and “defined data” (the data supplied by the application on DEFINE_DLC).

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific DLC, or to obtain the list information in several “chunks,” the **dlc_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

This list is ordered by the **dlc_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering).

If AP_LIST_FROM_NEXT is selected the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

VCB Structure

```
typedef struct query_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    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[2 ];        / reserved                      /
} DLC_DET_DATA;

typedef struct dlc_def_data
{
        unsigned  char  description[RD_LEN];
                                            / 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  reserv3[311];       / reserved                      /
        unsigned short  dlc_spec_data_len; / Length of DLC specific data  /
} DLC_DEF_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_DLC

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **dlc_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

dlc_name DLC name. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

dlc_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

dlc_summary.dlc_name
DLC name. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

dlc_summary.description
Resource description (as specified on DEFINE_DLC). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

dlc_summary.state
State of the DLC. This field is set to one of the following:

AP_ACTIVE
AP_NOT_ACTIVE
AP_PENDING_INACTIVE

dlc_summary.dlc_type
Type of DLC. Personal Communications supports the following types:

AP_LLC2
AP_SDLC
AP_TWINAX

dlc_detail.overlay_size
The number of bytes in this entry (including dlc_spec_data), and hence the offset to the next entry returned (if any).

dlc_detail.dlc_name
DLC name. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

dlc_detail.det_data.state
State of the DLC. This field is set to one of the following:

AP_ACTIVE
AP_NOT_ACTIVE
AP_PENDING_INACTIVE

dlc_detail.def_data.description
Resource description (as specified on DEFINE_DLC). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

dlc_detail.def_data.dlc_type

Type of DLC. Personal Communications supports the following types:

AP_LLC2
AP_SDLC
AP_TWINAX

dlc_detail.def_data.neg_ls_supp
Specifies whether the DLC supports negotiable link stations (AP_YES or AP_NO).

dlc_detail.def_data.port_types
Specifies the allowable port types for the supplied **dlc_type**. The value corresponds to one or more of the following Ored together.

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

This field will be set if **dlc_type** is set to a value other than AP_LLC2, AP_SDLC, or AP_TWINAX.

dlc_detail.def_data.dlc_spec_data_len
Unpadded length, in bytes, of data specific to the type of DLC. The data will be concatenated to the DLC_DETAIL structure. This data will be padded to end on a 4-byte boundary. This field should always be set to zero.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_DLC_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLUR_LU

QUERY_DLUR_LU returns a list of information about DLUR-supported LUs.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific LU, or to obtain the list information in several "chunks," the **lu_name** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered by the **lu_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering).

If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

The list of LUs returned may be filtered either by **pu_name** or by whether the LU is local or downstream or by both. If filtering by PU is desired, the **pu_name** field should be set (otherwise this field should be set to all zeros). If filtering by location is desired, the **filter** field should be set to AP_INTERNAL or AP_DOWNSTREAM (otherwise, if no filtering is required, this field should be set to AP_NONE). In addition to the detail information for each LU, an RSCV (Route Selection Control Vector) will be returned if this is specified by the **store_dlur_rscvs** field on the START_NODE parameters. This RSCV (specified in SNA Formats) defines the route through the network which the PLU-SLU session will take in a hop-by-hop form. This route will always terminate at the local node.

VCB Structure

```
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;          / local/downstream filter      /
} 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;
```

Note: The DLUR_LU_DETAIL overlay may be followed by a Route Selection Control Vector (RSCV) as defined by SNA Formats. This control vector defines the session route through the network and is carried on the BIND. The RSCV is included if the **store_dlur_rscvs** field on the START_NODE verb is set to AP_YES. If either the START_NODE **store_dlur_rscvs** field is set to AP_NO or the PLU-SLU session is not active then the **rscv_len** is set to zero.

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_DLUR_LU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **lu_name** specified (see below) represents an index value that is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

lu_name Name of LU being queried. This is an 8-byte alphanumeric type A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

pu_name PU name filter. This should be set to all zeros or an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If this field is set then only LUs associated with the specified PU are returned. This field is ignored if it is set to all zeros.

filter Location filter. Specifies whether the returned LUs should be filtered by location (AP_INTERNAL or AP_DOWNSTREAM). If no filter is required, this field should be set to AP_NONE.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

dlur_lu_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

dlur_lu_summary.lu_name
Name of LU. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

dlur_lu_detail.overlay_size
The number of bytes in this entry (including appended RSCV), and hence the offset to the next entry returned (if any).

dlur_lu_detail.lu_name
Name of LU. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

dlur_lu_detail.pu_name
Name of PU associated with the LU. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

dlur_lu_detail.dlus_name
Name of DLUS node if the SSCP-LU session is active. This is a 17-byte string composed of two type-A EBCDIC character strings

concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the SSCP-LU session is not active, this field will be set to all zeros.

dlur_lu_detail.lu_location
Location of LU. This is set to one of the following:

AP_INTERNAL
AP_DOWNSTREAM

dlur_lu_detail.nau_address
Network addressable unit address of the LU. This is in the range 1—2554.

dlur_lu_detail.plu_name
Name of PLU if the LU has an active PLU-SLU session. This is a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the PLU-SLU session is not active, this field will be set to all zeros.

dlur_lu_detail.rscv_len
Length of the RSCV which is appended to the **dlur_lu_detail** structure. (If none is appended, then the length is zero.) The RSCV will be padded to end on a 4-byte boundary.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_LU_NAME AP_INVALID_FILTER_OPTION AP_INVALID_LIST_OPTION

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLUR_PU

QUERY_DLUR_PU returns a list of information about DLUR-supported PUs.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific PU, or to obtain the list information in several "chunks," the **pu_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered by the **pu_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering).

If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

The list of PUs returned may be filtered either by **dlus_name** or by whether the PU is local or downstream or by both. If filtering by DLUS is desired, the **dlus_name** field should be set (otherwise this field should be set to all zeros). If filtering by PU location is desired, the **filter** field should be set to AP_INTERNAL or AP_DOWNSTREAM (otherwise, if no filtering is required, this field should be set to AP_NONE).

VCB Structure

```
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       /
} 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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_DLUR_PU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **pu_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

pu_name Name of PU being queried. This is an 8-byte alphanumeric type A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

dlus_name DLUS filter. This should be set to all zeros or to a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. If this field is set then only PUs associated with an SSCP-PU session to the specified DLUS node are returned. This field is ignored if it is set to all zeros.

filter This field should be set to AP_NONE.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

dlur_pu_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

dlur_pu_summary.pu_name
Name of PU. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

dlur_pu_summary.description
Resource description (as specified on DEFINE_INTERNAL_PU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

dlur_pu_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

dlur_pu_detail.pu_name
Name of PU. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

dlur_pu_detail.description
Resource description (as specified on DEFINE_INTERNAL_PU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

dlur_pu_detail.defined_dlus_name
Name of DLUS node defined by either a DEFINE_INTERNAL_PU verb or DEFINE_LS verb (with **dspu_services** set to AP_DLUR). This is a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

dlur_pu_detail.bkup_dlus_name
Name of backup DLUS node defined by either a DEFINE_INTERNAL_PU verb or DEFINE_LS verb (with **dspu_services** set to AP_DLUR). This is a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

dlur_pu_detail.pu_id
PU identifier defined in a DEFINE_INTERNAL_PU verb or obtained in an XID from a downstream PU. This a 4-byte hexadecimal string. Bits 0—11 are set to the Block number and bits 12—31 are set to the ID number that uniquely identifies the PU.

dlur_pu_detail.pu_location
Location of PU. This should be set to the following:

AP_INTERNAL

dlur_pu_detail.active_dlus_name
Name of DLUS node which the PU is currently using. This is a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, which is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the SSCP-PU session is not active, this field will be set to all zeros.

dlur_pu_detail.ans_support
Auto Network Shutdown support. This field is reserved if the SSCP-LU session is inactive. The support setting is sent to DLUR from the DLUS at SSCP-PU activation. It specifies whether link-level contact should be continued if the subarea node initiates an auto network shutdown procedure for the SSCP controlling the PU. This can be one of the following:

AP_CONT
AP_STOP

dlur_pu_detail.pu_status
Status of the PU (as seen by DLUR). This can be set to one of the following:

AP_RESET
The PU is in reset state.

AP_PEND_ACTPU
The PU is waiting for an ACTPU from the host.

AP_PEND_ACTPU_RSP
Having forwarded an ACTPU to the PU, DLUR is now waiting for the PU to respond to it.

AP_ACTIVE
The PU is active.

AP_PEND_DACTPU_RSP
Having forwarded a DACTPU to the PU, DLUR is waiting for the PU to respond to it.

AP_PEND_INOP
DLUR is waiting for all necessary events to complete before it deactivates the PU.

dlur_pu_detail.dlus_session_status
Status of the DLUS pipe currently being used by the PU. This can be one of the following:

AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE
AP_INACTIVE

dlur_pu_detail.fqpcid.pcid
Procedure Correlator ID used on the pipe. This is an 8-byte hexadecimal string. If the SSCP-PU session is not active this field will be set to zeros.

dlur_pu_detail.fqpcid.fqcp_name
Fully qualified Control Point name used on the pipe. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) If the SSCP-PU session is not active this field will be set to zeros.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_PU_NAME AP_INVALID_FILTER_OPTION AP_INVALID_LIST_OPTION

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLUS

QUERY_DLUS returns a list of information about DLUS nodes known by DLUR.

The information is returned as a list. To obtain information about a specific DLUS node, or to obtain the list information in several “chunks,” the **dlus_name** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

This list is ordered by the **dlus_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering).

If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

Note that this verb returns pipe statistics.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_DLUS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **dlus_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

dlus_name Name of DLUS being queried. This should be set to all zeros or a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, which is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

dlus_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

dlus_data.dlus_name
Name of DLUS. This is a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

dlus_data.is_default
Specifies whether the DLUS node has been designated as the default by a DEFINE_DLUR_DEFAULTS verb (AP_YES or AP_NO).

dlus_data.is_backup_default
Specifies whether the DLUS node has been designated as the backup default by a DEFINE_DLUR_DEFAULTS verb (AP_YES or AP_NO).

dlus_data.pipe_state
State of the pipe to the DLUS. It can have one of the following values:

AP_ACTIVE
AP_PENDING_ACTIVE
AP_INACTIVE
AP_PENDING_INACTIVE

dlus_data.num_active_pus
Number of PUs currently using the pipe to the DLUS.

dlus_data.pipe_stats.reqactpu_sent
Number of REQACTPUs sent to DLUS over the pipe.

dlus_data.pipe_stats.reqactpu_rsp_received
Number of RSP(REQACTPU)s received from DLUS over the pipe.

dlus_data.pipe_stats.actpu_received
Number of ACTPUs received from DLUS over the pipe.

dlus_data.pipe_stats.actpu_rsp_sent
Number of RSP(ACTPU)s sent to DLUS over the pipe.

dlus_data.pipe_stats.reqdactpu_sent
Number of REQDACTPUs sent to DLUS over the pipe.

dlus_data.pipe_stats.reqdactpu_rsp_received
Number of RSP(REQDACTPU)s received from DLUS over the pipe.

dlus_data.pipe_stats.dactpu_received
Number of DACTPUs received from DLUS over the pipe.

dlus_data.pipe_stats.dactpu_rsp_sent
Number of RSP(DACTPU)s sent to DLUS over the pipe.

dlus_data.pipe_stats.actlu_received
Number of ACTLUs received from DLUS over the pipe.

dlus_data.pipe_stats.actlu_rsp_sent
Number of RSP(ACTLU)s sent to DLUS over the pipe.

dlus_data.pipe_stats.dactlu_received
Number of DACTLUs received from DLUS over the pipe.

dlus_data.pipe_stats.dactlu_rsp_sent
Number of RSP(DACTLU)s sent to DLUS over the pipe.

dlus_data.pipe_stats.sscp_pu_mus_rcvd
Number of SSCP-PU MUs received from DLUS over the pipe.

dlus_data.pipe_stats.sscp_pu_mus_sent
Number of SSCP-PU MUs sent to DLUS over the pipe.

dlus_data.pipe_stats.sscp_lu_mus_rcvd
Number of SSCP-LU MUs received from DLUS over the pipe.

dlus_data.pipe_stats.sscp_lu_mus_sent
Number of SSCP-LU MUs sent to DLUS over the pipe.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_DLUS_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_FOCAL_POINT

QUERY_FOCAL_POINT returns information about focal points that Personal Communications knows about.

This information is returned as a list. To obtain information about a specific focal point category or to obtain the list information in several “chunks,” the **ms_category** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

VCB Structure

```
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[2 ];            / reserved                          /
        unsigned short   number_of_appls;        / number of applications            /
} FP_DATA;
```

Each **fp_data** is followed by **number_of_appls** application names. Each application name has the following format.

```
typedef struct application_id
{
        unsigned char    appl_name[8];      / application name                /
} APPLICATION_ID;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_FOCAL_POINT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information: The **ms_category** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

ms_category Management Services category. This may either be one of the four-byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services categories as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation defined name. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
The number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

fp_data.overlay_size
The number of bytes in this entry (including any application names, and hence the offset to the next entry returned (if any)).

fp_data.ms_appl_name
Name of the currently active focal point application. This may either be one of the four-byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services applications as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation defined name.

fp_data.ms_category
Management Services category. This may either be one of the four byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services categories as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation defined name.

fp_data.description
Resource description (as specified on DEFINE_FOCAL_POINT). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

fp_data.fp_fqcp_name
Currently active focal point's fully qualified control point name. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

fp_data.bkup_appl_name
Name of backup focal point application. This may either be one of the four-byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services applications as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation defined name.

fp_data.bkup_fp_fqcp_name
Backup focal point's fully qualified control point name. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

fp_data.implicit_appl_name
Name of implicit focal point application (specified using the DEFINE_FOCAL_POINT verb). This may either be one of the four byte architecturally defined values (right-padded with EBCDIC spaces) for

Management Services applications as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation defined name. This field will be the same as the **ms_appl_name** if the implicit focal point is the currently active focal point.

fp_data.bkup_fp_fqcp_name
Implicit focal point's fully qualified control point name (as specified using the DEFINE_FOCAL_POINT verb). This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field will be the same as the **fp_fqcp_name** if the implicit focal point is the currently active focal point.

fp_data.fp_type
Type of focal point. Refer to SNA Management Services for further detail. This will be one of the following:

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
Status of the focal point. This can be one of the following:

AP_NOT_ACTIVE
The focal point is currently not active.

AP_ACTIVE
The focal point is currently active.

AP_PENDING
The focal point is pending active. This occurs after an implicit request has been sent to the focal point and before the response has been received.

AP_NEVER_ACTIVE
No focal point information is available for the specified category although application registrations for the category have been accepted.

fp_data.fp_routing
Type of routing that applications should specify when using MDS transport to send data to the focal point.

AP_DEFAULT
Default routing is used to deliver the MDS_MU to the focal point.

AP_DIRECT
The MDS_MU will be routed on a session directly to the focal point.

fp_data.number_of_appls
Number of applications registered for this focal point category. Following this field will be **number_of_appls application_id entries**, one for each application registered for the focal point category.

fp_data.appl_name
Name of application registered for focal point category. This may either be one of the four-byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services applications as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation defined name.

Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_MS_CATEGORY AP_INVALID_LIST_OPTION

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LOCAL_LU

QUERY_LOCAL_LU returns information about local LUs. QUERY_LOCAL_LU can be issued to retrieve information about the Personal Communications CP LU.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific local LU, or to obtain the list information in several "chunks," the **lu_name** or **lu_alias** field should be set. If the **lu_name** field is nonzero it will be used to determine the index. If the **lu_name** field is set to all zeros, the **lu_alias** will be used to determine the index. If both the **lu_name** and the **lu_alias** fields are set to all zeros then the LU associated with the CP (the default LU) will be used. If the **list_options** field is set to AP_FIRST_IN_LIST then both of these fields will be ignored. (In this case, the returned list will be ordered by LU alias if the AP_LIST_BY_ALIAS **list_options** is set, otherwise it will be ordered by LU name). See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered on either **lu_alias** or **lu_name** according to the options specified. The field is ordered by EBCDIC lexicographical ordering.

The list of local LUs returned may be filtered by the name of the PU that they are associated with. In this case, the **pu_name** field should be set (otherwise this field should be set to all zeros).

VCB Structure

```
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[RD_LEN];
                                                / resource description               /
} LOCAL_LU_SUMMARY;

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_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    reserv1;             / reserved                   /
        unsigned  char    reserv2;             / reserved                   /
        unsigned char     pu_name[8];          / PU name                    /
        unsigned  char    reserv3[8];          / reserved                   /
        unsigned  char    attach_routing_data[128];
                                               / routing data for           /
                                               / incoming attaches          /
} LOCAL_LU_DEF_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_LOCAL_LU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB, in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **lu_name** (or **lu_alias** if the **lu_name** is set to all zeros) specified represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

AP_LIST_BY_ALIAS
The returned list is ordered by **lu_alias**. This option is only valid when AP_FIRST_IN_LIST is specified. If AP_LIST_FROM_NEXT or AP_LIST_INCLUSIVE is specified, the list ordering will depend on whether an **lu_name** or **lu_alias** has been supplied as a starting point.

lu_name LU name. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the index. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

lu_alias Locally defined LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If both the **lu_name** and the **lu_alias** field are set to all zeros, the LU associated with the CP (the default LU) is used. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

pu_name PU name filter. This should be set to all zeros or an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If this field is set then only Local LUs associated with this PU are returned. This field is ignored if it is set to all zeros.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

local_lu_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

local_lu_summary.lu_name
LU name. This name is an 8-byte type-A EBCDIC character string.

local_lu_summary.lu_alias
Locally defined LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

local_lu_summary.description
Resource description (as specified on DEFINE_LOCAL_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

local_lu_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

local_lu_detail.lu_name
LU name. This name is an 8-byte type-A EBCDIC character string.

local_lu_detail.def_data.description
Resource description (as specified on DEFINE_LOCAL_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

local_lu_detail.def_data.lu_alias
Locally defined LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

local_lu_detail.def_data.nau_address
Network addressable unit address of the LU which is in the range 0—2554. A nonzero value implies the LU is a dependent LU. Zero implies the LU is an independent LU.

local_lu_detail.def_data.syncpt_support
Reserved.

local_lu_detail.def_data.lu_session_limit
Maximum number of sessions for the local LU. A value of zero indicates that there is no limit.

local_lu_detail.def_data.default_pool
AP_YES if the LU is a member of the dependent LU6.2 default pool. Always AP_NO for independent LUs.

local_lu_detail.def_data.pu_name
Name of the PU which this LU will use. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is only used by dependent LUs, and will be set to all binary zeros for independent LUs.

local_lu_detail.def_data.attach_routing_data
Reserved.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_LU_ALIAS AP_INVALID_LU_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LOCAL_TOPOLOGY

All APPN nodes maintain a local topology database that holds information about the transmission groups (TGs) to all adjacent nodes.

QUERY_LOCAL_TOPOLOGY allows information about these TGs to be returned.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific local TG, or to obtain the list information in several "chunks," the **dest**, **dest_type**, and **tg_num fields** should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), these fields will be ignored. See "Querying the Node" on page 12, for background on how the list formats are used. This list is ordered on **dest** first, then on **dest_type** and finally on **tg_num**. The **dest** name is ordered by name length first, then by lexicographical ordering for names of the same length. The **dest_type** field follows the order: AP_LEN_NODE, AP_NETWORK_NODE, AP_END_NODE, AP_VRN. The **tg_num** is ordered numerically.

If AP_LIST_INCLUSIVE is selected, the returned list starts from the first valid record of that name.

If AP_LIST_FROM_NEXT is selected, the list will begin from the first valid record with a name following the one specified.

VCB Structure

```
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    reserva[16];        /  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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_LOCAL_TOPOLOGY

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The combination of the **dest**, **dest_type** and **tg_num** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

dest Fully qualified destination node name for the TG. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

dest_type Node type of the destination node for this TG. This can be one of the following:

AP_NETWORK_NODE
AP_VRN
AP_END_NODE
AP_LEARN_NODE

If the **dest_type** is unknown, AP_LEARN_NODE must be specified. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

tg_num Number associated with the TG. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

local_topology_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

local_topology_summary.dest
Fully qualified destination node name for the TG. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

local_topology_summary.dest_type
Type of the destination node for this TG. This is set to one of the following:

AP_NETWORK_NODE
AP_VRN
AP_END_NODE

Note that if **dest_type** is set to AP_END_NODE, this specifies that the TG destination is either to a LEN node or to an end node.

local_topology_summary.tg_num
Number associated with the TG.

local_topology_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

local_topology_detail.dest
Fully qualified destination node name for the TG. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

local_topology_detail.dest_type
Type of the destination node for this TG. This is set to one of the following:

AP_NETWORK_NODE
AP_VRN
AP_END_NODE

Note that if **dest_type** is set to AP_END_NODE, this specifies that the TG destination is either to a LEN node or to an end node.

local_topology_detail.tg_num
Number associated with the TG.

local_topology_detail.dlc_data.length
Length of DLC address of connection to a VRN (set to zero if **dest_type** is not AP_VRN).

local_topology_detail.dlc_data.address
DLC address of connection to VRN.

local_topology_detail.rsn
Resource Sequence Number. This is assigned by the Network Node that owns this resource.

local_topology_detail.status
Specifies the status of the TG. This may be one or more of the following ORed together:

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 characteristics (See "DEFINE_CN" on page 31).

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_TG AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LS

QUERY_LS returns a list of information about the link stations defined at the node. This information is structured as “determined data” (data gathered dynamically during execution) and “defined data” (the data supplied by the application on DEFINE_LS).

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific LS, or to obtain the list information in several “chunks,” the **ls_name** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

This list is ordered by the **ls_name**. Ordering is according to name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with IBM's 6611 APPN MIB ordering). If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

The list of link stations returned may be filtered by the name of the port that they belong to. In this case, the **port_name** field should be set (otherwise this field should be set to all zeros).

VCB Structure

```
typedef struct query_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    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  char   reservb[32];           / reserved                           /
} LS_DET_DATA;

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    reserv3[35];            /   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&lbraket.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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_LS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **ls_name** specified (see below) represents an index value that is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

ls_name Link station name. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

port_name Port name filter. This should be set to all zeros or an 8-byte alphanumeric type A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If this field is set then only link stations belonging to this port are returned. This field is ignored if it is set to all zeros.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

ls_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

ls_summary.ls_name
Name of link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

ls_summary.description
Resource description (as specified on DEFINE_LS). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

ls_summary.dlc_type
Type of DLC. Personal Communications supports the following types:

AP_LLC2
AP_SDLC
AP_TWINAX

ls_summary.state
State of this link station. This field is set to one of the following:

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

ls_summary.act_sess_count
The total number of active sessions (both endpoint and intermediate) using the link.

ls_summary.det_adj_cp_name
Fully-qualified, 17-byte, adjacent CP name determined during link activation. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This will be null if the LS is inactive.

ls_summary.det_adj_cp_type
Type of the adjacent node determined during link activation. It is one of the following:

AP_END_NODE
AP_NETWORK_NODE
AP_LEARN_NODE
AP_VRN

This will be AP_LEARN_NODE if the LS is inactive.

ls_summary.port_name
Name of port associated with this link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

ls_summary.adj_cp_name
Fully-qualified, 17-byte, adjacent CP name composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, which is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This will be null for an implicit link.

ls_summary.adj_cp_type
Type of the adjacent node. It is one of the following:

AP_END_NODE
AP_NETWORK_NODE
AP_LEARN_NODE

ls_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

ls_detail.ls_name
Name of link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

ls_detail.det_data.act_sess_count
Total number of active sessions (both endpoint and intermediate) using the link.

ls_detail.det_data.dlc_type
Type of DLC. Personal Communications supports the following types:

AP_LLC2
AP_SDLC
AP_TWINAX

Additional DLC types may be defined by specifying the new type on the DEFINE_DLC verb. See “DEFINE_DLC” on page 45, for more information.

ls_detail.det_data.state
State of this link station. This field is set to one of the following:

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

ls_detail.det_data.sub_state
This field provides more detailed information about the state of this link station. This field is set to one of the following:

AP_SENT_CONNECT_OUT
AP_PENDING_XID_EXCHANGE
AP_SENT_ACTIVATE_AS
AP_SENT_SET_MODE
AP_ACTIVE
AP_SENT_DEACTIVATE_AS_ORDERLY
AP_SENT_DISCONNECT
AP_WAITING_STATS
AP_RESET

ls_detail.det_data.det_adj_cp_name
Fully-qualified, 17-byte, adjacent CP name determined during link activation. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

ls_detail.det_data.det_adj_cp_type
Type of the adjacent node determined during link activation. It is one of the following:

AP_END_NODE
AP_NETWORK_NODE
AP_LEARN_NODE
AP_VRN

ls_detail.det_data.dlc_name
Name of the DLC. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

ls_detail.det_data.dynamic
Specifies whether the link was defined explicitly (by a DEFINE_LS command), or implicitly or dynamically (either in response to a connection request from the adjacent node, or to connect dynamically to another node across a Connection Network). This may be AP_YES or AP_NO.

ls_detail.det_data.migration
Specifies whether the adjacent node is a migration level node (such as a Low Entry Networking (LEN) node), or a full APPN Network Node or End Node (AP_YES, AP_NO, or AP_UNKNOWN).

ls_detail.det_data.tg_num
Number associated with the TG.

ls_detail.det_data.ls_stats.in_xid_bytes
Total number of XID (Exchange Identification) bytes received on this link station.

ls_detail.det_data.ls_stats.in_msg_bytes
Total number of data bytes received on this link station.

ls_detail.det_data.ls_stats.in_xid_frames
Total number of XID (Exchange Identification) frames received on this link station.

ls_detail.det_data.ls_stats.in_msg_frames
Total number of data frames received on this link station.

ls_detail.det_data.ls_stats.out_xid_bytes
Total number of XID (Exchange Identification) bytes sent on this link station.

ls_detail.det_data.ls_stats.out_msg_bytes
Total number of data bytes sent on this link station.

ls_detail.det_data.ls_stats.out_xid_frames
Total number of XID (Exchange Identification) frames sent on this link station.

ls_detail.det_data.ls_stats.out_msg_frames
Total number of data frames sent on this link station.

ls_detail.det_data.ls_stats.in_invalid_sna_frames
Total number of SNA invalid frames received on this link station.

ls_detail.det_data.ls_stats.in_session_control_frames
Total number of session control frames received on this link station.

ls_detail.det_data.ls_stats.out_session_control_frames
Total number of session control frames sent on this link station.

ls_detail.det_data.ls_stats.echo_rsps
Number of echo responses received from the adjacent node. Echo requests are sent periodically to gauge the propagation delay to the adjacent node.

ls_detail.det_data.ls_stats.current_delay
Time (in milliseconds) that it took for the last test signal to be sent and returned from this link station to the adjacent link station.

ls_detail.det_data.ls_stats.max_delay
Longest time taken (in milliseconds) for a test signal to be sent and returned from this link station to the adjacent link station.

ls_detail.det_data.ls_stats.min_delay
Shortest time taken (in milliseconds) for a test signal to be sent and returned from this link station to the adjacent link station.

ls_detail.det_data.ls_stats.max_delay_time
Time since system startup (in hundredths of a second) when the longest delay occurred.

ls_detail.det_data.ls_stats.good_xids
Total number of successful XID exchanges that have occurred on this link station since it was started.

ls_detail.det_data.ls_stats.bad_xids
Total number of unsuccessful XID exchanges that have occurred on this link station since it was started.

ls_detail.det_data.start_time
Time since system startup (in hundredths of a second) when the link station was last activated (that is, the mode setting commands completed).

ls_detail.det_data.stop_time
Time since system startup (in hundredths of a second) when the link station was last deactivated.

ls_detail.det_data.up_time
The total time (in hundredths of a second) that this link station has been active since system startup.

ls_detail.det_data.current_state_time
The total time (in hundredths of a second) that this link station has been in the current state.

ls_detail.det_data.deact_cause
The cause of the last deactivation of the link station. The field is set to one of the following:

AP_NONE
The link station has never been deactivated.

AP_DEACT_OPER_ORDERLY
The link station was deactivated as a result of an orderly STOP command from an operator.

AP_DEACT_OPER_IMMEDIATE
The link station was deactivated as a result of an immediate STOP command from an operator.

AP_DEACT_AUTOMATIC
The link station was deactivated automatically, for example because there were no more sessions using the link station.

AP_DEACT_FAILURE
The link station was deactivated because of a failure.

ls_detail.det_data.hpr_support
The level of high-performance routing (HPR) supported on this TG (AP_NONE, AP_BASE or AP_RTP), taking account of the capabilities of the local and adjacent nodes.

ls_detail.det_data.anr_label
The HPR automatic network routing (ANR) label allocated to the local link.

ls_detail.det_data.hpr_link_lvl_error
Specifies whether link-level error recovery is being used for HPR traffic on the link.

ls_detail.det_data.ls_role
Specifies the link station role that this link station has assumed after negotiation with the partner link station.

ls_detail.det_data.node_id
Node ID received from adjacent node during XID exchange. This a 4-byte hexadecimal string.

ls_detail.def_data.description
Resource description (as specified on DEFINE_LS). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

ls_detail.def_data.port_name
Name of port associated with this link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant. If the link is to a VRN, this field specifies the name of the actual port used to connect to the VRN (as specified in the DEFINE_CN verb).

ls_detail.def_data.adj_cp_name
Fully-qualified 17-byte adjacent CP name which is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This is defined if **back_lvl_len_end_node** is not set to AP_NO, or if the port associated with the link station is defined to be switched.

ls_detail.def_data.adj_cp_type
Adjacent node type.

AP_END_NODE
Specifies that the node is a LEN or End node.

AP_NETWORK_NODE
Specifies that the node is a network node.

AP_LEARN_NODE
Specifies that the node type will be learned during XID exchange Note that a Link Station to a VRN is always dynamic and is therefore not defined.

ls_detail.def_data.dest_address.length
Length of destination link station's address on adjacent node.

ls_detail.def_data.dest_address.address
Link station's destination address on adjacent node.

ls_detail.def_data.auto_act_supp
Specifies whether the link will be activated automatically after it has been started by a START_LS verb, and stopped by a STOP_LS. (AP_YES or AP_NO).

ls_detail.def_data.tg_number
Pre-assigned TG number (in the range one to 20). This number is used to represent the link when the link is activated. Zero indicates that the TG number is not pre-assigned but is negotiated when the link is activated.

ls_detail.def_data.limited_resource
Specifies whether this link station is to be deactivated when there are no sessions using the link. This is set to one of the following:

AP_NO
The link is not a limited resource and will not be deactivated automatically.

AP_YES or AP_NO_SESSIONS
The link is a limited resource and will be deactivated automatically when no active sessions are using it.

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

ls_detail.def_data.solicit_sscp_sessions
AP_YES requests the host to initiate sessions between the SSCP and the local CP and dependent LUs. AP_NO requests no sessions with the SSCP on this link.

ls_detail.def_data.pu_name
Name of local PU which is going to use this link if **solicit_sscp_sessions** is set to AP_YES. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If **solicit_sscp_sessions** is set to AP_NO, this field is reserved.

ls_detail.def_data.disable_remote.act
Specifies whether remote activation of this link is supported (AP_YES or AP_NO).

ls_detail.def_data.dspu_services
Reserved.

ls_detail.def_data.dspu_name
Reserved.

ls_detail.def_data.dlus_name
Reserved.

ls_detail.def_data.bkup_dlus_name
Reserved.

ls_detail.def_data.hpr_supported
Specifies whether HPR is supported on this link (AP_YES or AP_NO).

ls_detail.def_data.hpr_link_lvl_error
Specifies whether the HPR link-level error recovery tower is supported on this link (AP_YES or AP_NO). Note that the parameter is reserved if **hpr_supported** is set to AP_NO.

ls_detail.def_data.link_deact_timer
Limited resource link deactivation timer (in seconds). A limited resource link is automatically deactivated if no data traverses the link for the duration of this timer, and no sessions are using the link.) This parameter is reserved if **limited_resource** is set to AP_NO.

ls_detail.def_data.default_nn_server
Specifies whether a link can be automatically activated by an end node to support CP-CP sessions to a Network Node Server. (AP_YES or AP_NO). The link must be defined to support CP-CP sessions for this field to take effect.

ls_detail.def_data.ls_attributes
Specifies further information about the adjacent node.

ls_detail.def_data.ls_attributes[0]
Host type.

AP_SNA
Standard SNA host.

AP_FNA
FNA (VTAM-F) host.

AP_HNA
HNA host.

ls_detail.def_data.adj_node_id
Defined node ID of adjacent node.

ls_detail.def_data.local_node_id
Node ID sent in XIDs on this Link Station. This a 4-byte hexadecimal string. If this field is set to zero, the **node_id** specified on the START_NODE signal is used in XID exchanges. If this field is nonzero, it replaces the value specified on START_NODE for XID exchanges on this LS.

ls_detail.def_data.cp_cp_sess_support
Specifies whether CP-CP sessions are supported (AP_YES or AP_NO).

ls_detail_def_data.use_default_tg_chars
Specifies whether the TG characteristics supplied on the DEFINE_LS were discarded in favor of the default characteristics supplied on the DEFINE_PORT (AP_YES or AP_NO). This field does not apply to implicit links.

ls_detail.def_data.tg_chars
TG characteristics (See “DEFINE_CN” on page 31).

ls_detail.def_data.target_pacing_count
Numeric value between 1 and 32767 inclusive indicating the desired pacing window size for BINDs on this TG. The number is only significant when fixed bind pacing is being performed. Note that Personal Communications does not currently use this value.

ls_detail.def_data.max_send_btu_size
Maximum BTU size that can be sent.

ls_detail.def_data.ls_role
The link station role that this link station should assume. This may be any one of AP_LS_NEG, AP_LS_PRI or AP_LS_SEC to select a role of negotiable, primary or secondary. The field may also be set to AP_USE_PORT_DEFAULTS to select the value configured on the DEFINE_PORT verb.

ls_detail.def_data.link_spec_data_len
Unpadded length, in bytes, of data passed unchanged to link station component during initialization. The data is concatenated to the LS_DETAIL structure. This data will be padded to end on a 4-byte boundary.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_LINK_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LU_0_TO_3

QUERY_LU_0_TO_3 returns information about local LUs of type 0, 1, 2, or 3. This information is structured as “determined data” (data gathered dynamically during execution) and “defined data” (the data supplied by the application on DEFINE_LU_0_TO_3).

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific local LU, or to obtain the list information in several “chunks,” the **lu_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored.

VCB Structure

```
typedef struct query_lu_ _to_3
{
        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];              / PU name filter                      /
        unsigned char    lu_name[8];              / LU name                             /
        unsigned  char   host_attachment;         / Host  attachment  filter            /
} QUERY_LU_ _TO_3;

typedef struct lu_ _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_ _TO_3_SUMMARY;

typedef struct lu_ _to_3_detail
{
        unsigned short   overlay_size;            / size of this entry                  /
        unsigned char    lu_name[8];              / LU name                             /
        unsigned  char   reserv1[2];              / reserved                            /
        LU_ _TO_3_DET_DATA det_data;              / Determined data                     /
        LU_ _TO_3_DEF_DATA def_data;              / Defined data                        /
} LU_ _TO_3_DETAIL;
```

```
typedef struct lu_ _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     reserva[192 ];      / reserved                        /
} LU_ _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     reserve;            / reserved                       /
} SESSION_STATS;

typedef struct lu_ _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     reserv2[8];         / reserved                       /
        unsigned  char     app_spec_def_data[16];
                                        / Application Specified Data       /
} LU_ _TO_3_DEF_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_LU_0_TO_3

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **lu_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

lu_name Name of the local LU that is being queried. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

pu_name PU name. Only LUs which use this PU will be returned. If a list of all LUs is required then this field should be set to all binary zeros.

host_attachment Filter for host attachment.

AP_NONE
Return information about all LUs.

AP_DLUR_ATTACHED
Return information about all LUs which are supported by DLUR.

AP_DIRECT_ATTACHED
Return information about only those LUs which are directly attached to the host system.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

lu_0_to_3_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

lu_0_to_3_summary.pu_name
Name of local PU which this LU is using. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

lu_0_to_3_summary.lu_name
Name of the local LU that is being queried. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

lu_0_to_3_summary.description
Resource description (as specified on DEFINE_LU_0_TO_3). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

lu_0_to_3_summary.nau_address
Network addressable unit address of the LU which is in the range 1—2554.

lu_0_to_3_summary.lu_sscp_sess_active
Specifies whether the LU-SSCP session is active (AP_YES or AP_NO).

lu_0_to_3_summary.appl_conn_active
Specifies whether an application is using the LU (AP_YES or AP_NO).

lu_0_to_3_summary.plu_sess_active
Specifies whether the PLU-SLU session is active (AP_YES or AP_NO).

lu_0_to_3_summary.host_attachment
LU host attachment type:

AP_DLUR_ATTACHED
LU is attached to host system using DLUR.

AP_DIRECT_ATTACHED
LU is directly attached to host system.

lu_0_to_3_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

lu_0_to_3_detail.lu_name
Name of the local LU that is being queried. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

lu_0_to_3_detail.det_data.lu_sscp_sess_active
Specifies whether the LU-SSCP session is active (AP_YES or AP_NO).

lu_0_to_3_detail.det_data.appl_conn_active
Specifies whether this LU is currently being used by an application (AP_YES or AP_NO).

lu_0_to_3_detail.det_data.plu_sess_active
Specifies whether the PLU-SLU session is active (AP_YES or AP_NO).

lu_0_to_3_detail.det_data.host_attachment
LU host attachment type:

AP_DLUR_ATTACHED
LU is attached to host system using DLUR.

AP_DIRECT_ATTACHED
LU is directly attached to host system.

lu_0_to_3_detail.det_data.lu_sscp_stats.rcv_ru_size
This field is always reserved.

lu_0_to_3_detail.det_data.lu_sscp_stats.send_ru_size
This field is always reserved.

lu_0_to_3_detail.det_data.lu_sscp_stats.max_send_btu_size
Maximum BTU size that can be sent.

lu_0_to_3_detail.det_data.lu_sscp_stats.max_rcv_btu_size
Maximum BTU size that can be received.

lu_0_to_3_detail.det_data.lu_sscp_stats.max_send_pac_win
This field will always be set to zero.

lu_0_to_3_detail.det_data.lu_sscp_stats.cur_send_pac_win
This field will always be set to zero.

lu_0_to_3_detail.det_data.lu_sscp_stats.max_rcv_pac_win
This field will always be set to zero.

lu_0_to_3_detail.det_data.lu_sscp_stats.cur_rcv_pac_win
This field will always be set to zero.

lu_0_to_3_detail.det_data.lu_sscp_stats.send_data_frames
Number of normal flow data frames sent.

lu_0_to_3_detail.det_data.lu_sscp_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

lu_0_to_3_detail.det_data.lu_sscp_stats.send_data_bytes
Number of normal flow data bytes sent.

lu _0_to_3_detail.det_data.lu_sscp_stats.rcv_data_frames
Number of normal flow data frames received.

lu_0_to_3_detail.det_data.lu_sscp_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

lu_0_to_3_detail.det_data.lu_sscp_stats.rcv_data_bytes
Number of normal flow data bytes received.

lu_0_to_3_detail.det_data.lu_sscp_stats.sidh
Session ID high byte.

lu_0_to_3_detail.det_data.lu_sscp_stats.sidl
Session ID low byte.

lu_0_to_3_detail.det_data.lu_sscp_stats.odai
Origin Destination Address Indicator. When bringing up a session, the sender of the ACTLU sets this field to zero if the local node contains the primary link station, and sets it to one if the ACTLU sender is the node containing the secondary link station.

lu_0_to_3_detail.det_data.lu_sscp_stats.ls_name
Link station name associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant. This field can be used to correlate this session with the link over which the session flows.

lu_0_to_3_detail.det_data.plu_stats.rcv_ru_size
Maximum receive RU size.

lu_0_to_3_detail.det_data.plu_stats.send_ru_size
Maximum send RU size.

lu_0_to_3_detail.det_data.plu_stats.max_send_btu_size
Maximum BTU size that can be sent.

lu_0_to_3_detail.det_data.plu_stats.max_rcv_btu_size
Maximum BTU size that can be received.

lu_0_to_3_detail.det_data.plu_stats.max_send_pac_win
Maximum size of the send pacing window on this session.

lu_0_to_3_detail.det_data.plu_stats.cur_send_pac_win
Current size of the send pacing window on this session.

lu_0_to_3_detail.det_data.plu_stats.max_rcv_pac_win
Maximum size of the receive pacing window on this session.

lu_0_to_3_detail.det_data.plu_stats.cur_rcv_pac_win
Current size of the receive pacing window on this session.

lu_0_to_3_detail.det_data.plu_stats.send_data_frames
Number of normal flow data frames sent.

lu_0_to_3_detail.det_data.plu_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

lu_0_to_3_detail.det_data.plu_stats.send_data_bytes
Number of normal flow data bytes sent.

lu_0_to_3_detail.det_data.plu_stats.rcv_data_frames
Number of normal flow data frames received.

lu_0_to_3_detail.det_data.plu_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

lu_0_to_3_detail.det_data.plu_stats.rcv_data_bytes
Number of normal flow data bytes received.

lu_0_to_3_detail.det_data.plu_stats.sidh
Session ID high byte.

lu_0_to_3_detail.det_data.plu_stats.sidl
Session ID low byte.

lu_0_to_3_detail.det_data.plu_stats.odai
Origin Destination Address Indicator. When bringing up a session, the sender of the BIND sets this field to zero if the local node contains the primary link station, and sets it to one if the BIND sender is the node containing the secondary link station.

lu_0_to_3_detail.det_data.plu_stats.ls_name
Link station name associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

lu_0_to_3_detail.det_data.plu_name
Primary LU name. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. (If the PLU-SLU session is inactive this field is reserved).

lu_0_to_3_detail.det_data.session_id
Eight byte internal identifier of the PLU-SLU session.

lu_0_to_3_detail.det_data.app_spec_det_data
Reserved.

lu_0_to_3_detail.det_data.app_type
Reserved.

lu_0_to_3_detail.def_data.description
Resource description (as specified on DEFINE_LU_0_TO_3). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

lu_0_to_3_detail.def_data.nau_address
Network addressable unit address of the LU which is in the range 1—2554.

lu_0_to_3_detail.def_data.pool_name
Name of pool to which this LU belongs. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If the LU does not belong to a pool, this field is set to all binary zeros.

lu_0_to_3_detail.def_data.pu_name
Name of the PU (as specified on the DEFINE_LS verb) which this LU will use. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

lu_0_to_3_detail.def_data.priority
LU priority when sending to the host. This is set to one of the following:

AP_NETWORK
AP_HIGH
AP_MEDIUM
AP_LOW

lu_0_to_3_detail.def_data.lu_model
Model type and number of the LU. This is set to one of the following:

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_UNKNOWN

lu_0_to_3_detail.def_data.app_spec_def_data
Application specified data from DEFINE_LU_0_TO_3. Personal Communications does not interpret this field, it is simply stored and returned on the QUERY_LU_0_TO_3 verb.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_LU_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MDS_APPLICATION

QUERY_MDS_APPLICATION returns a list of applications which have registered for MDS level messages.

Applications register by issuing the REGISTER_MS_APPLICATION verb described in Chapter 14, “Management Services Verbs” on page 389.

To obtain information about a specific application, or to obtain the list information in several “chunks,” the **application** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

VCB Structure

```
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[2 ];             / reserved                         /
} MDS_APPLICATION_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_MDS_APPLICATION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information: The **application** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

application Application name. The name is an 8-byte alphanumeric type-A EBCDIC character string. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
The number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

mds_application_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

mds_application_data.application
Name of registered application. The name is an 8-byte alphanumeric type-A EBCDIC character string.

mds_application_data.max_rcv_size
The maximum number of bytes that the application can receive in one chunk (this is specified when an application registers with MDS). For more information about MDS-level application registration refer to Chapter 14, “Management Services Verbs.”

Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_APPLICATION_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc	AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MDS_STATISTICS

QUERY_MDS_STATISTICS returns Management Services statistics. This verb can be used to gauge the level of MDS routing traffic. The information can also be used to tune the size of the send alert queue which is configurable at node initialization time.

VCB Structure

```
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[2 ];             / reserved                         /
} QUERY_MDS_STATISTICS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_MDS_STATISTICS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

alerts_sent
Number of locally originated alerts sent using the MDS transport system.

alert_errors_rcvd
Number of error messages received by MDS indicating a delivery failure for a message containing an alert.

uncorrelated_errors_rcvd
Number of error messages received by MDS indicating a delivery failure for a message containing an alert. Delivery failure occurs when the error message could not be correlated to an alert on the MDS send alert queue. MDS maintains a fixed-size queue where it caches alerts sent to the problem determination focal point. Once the queue reaches maximum size, the oldest alert will be discarded and replaced by the new alert. If a delivery error message is received, MDS attempts to correlate the error message to a cached alert so that the alert may be held until the problem determination focal point is restored.

Note: The two counts, **alert_errors_rcvd** and **uncorrelated_errors_rcvd** are maintained such that the size of the send alert queue (specified on the START_NODE) can be tuned. An increasing **uncorrelated_errors_rcvd** over time indicates that the send alert queue size is too small.

mds_mus_rcvd_local
Number of MDS_MUs received from local applications.

mds_mus_rcvd_remote
Number of MDS_MUs received from remote nodes using the MDS_RECEIVE and MSU_HANDLER transaction programs.

mds_mus_delivered_local
Number of MDS_MUs successfully delivered to local applications.

mds_mus_delivered_remote
Number of MDS_MUs successfully delivered to a remote node using the MDS_SEND transaction program.

parse_errors
Number of MDS_MUs received which contained header format errors.

failed_deliveries
Number of MDS_MUs this node failed to deliver.

ds_searches_performed
Reserved.

unverified_errors
Reserved.

Returned Parameters: Function Support Check

If the verb does not execute because the relevant START_NODE parameter(s) were not set, Personal Communications returns the following parameter:

primary_rc AP_FUNCTION_NOT_SUPPORTED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MODE

QUERY_MODE returns information about modes that have been used by a local LU with a particular partner LU. The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific mode, or to obtain the list information in several "chunks," the **mode_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. Note that the **lu_name** (or **lu_alias**) and **plu_alias** (or **fqplu_name**) fields must always be set. The **lu_name**, if nonzero, will be used in preference to the **lu_alias**. See "Querying the Node" on page 12, for background on how the list formats are used.

The list only includes information for the local LU specified by the **lu_name** (or **lu_alias**). This list is ordered by the **fqplu_name** followed by the **mode_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering).

If **plu_alias** is set to all zeros, the **fqplu_name** value will be used, otherwise the **plu_alias** is always used and the **fqplu_name** is ignored.

The list of modes returned may be filtered according to whether they currently have any active sessions. If filtering is desired, the **active_sessions** field should be set to AP_YES (otherwise this field should be set to AP_NO). This verb returns information that is determined once the mode begins to be used by a local LU with a partner LU. The QUERY_MODE_DEFINITION verb returns definition information only.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_MODE

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The combination of **lu_name** (or **lu_alias** if the **lu_name** is set to all zeros), **plu_alias** (or **fqplu_name** if the **plu_alias** is set to all zeros) and **mode_name** specified (see below) represents an index value that is used to specify the starting point of the actual information to be returned. When a partner LU index is specified, information about other partner LUs is included in the list, if possible.

AP_FIRST_IN_LIST
If **plu_alias** and **fqplu_name** are set to all zeros, the returned list starts from the first partner LU in the list, and the **mode_name** index is ignored. If either **plu_alias** or **fqplu_name** is specified, the list starts at this index, but the **mode_name** index value is ignored, and the returned list starts from the first mode entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

lu_name LU name. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the index.

lu_alias Locally defined LU alias. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_name** and the **lu_alias** are set to all zeros then the LU associated with the CP (the default LU) is used.

plu_alias Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If this field is set to all zeros, the **fqplu_name** field will be used for determining the index.

fqplu_name 17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

mode_name Mode name which designates the network properties for a group of sessions. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

active_sessions Active session filter. Specifies whether the returned modes should be filtered according to whether they currently have any active sessions (AP_YES or AP_NO).

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

mode_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

mode_summary.mode_name
Mode name which designates the network properties for a group of sessions. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

mode_summary.description
Resource description (as specified on DEFINE_MODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

mode_summary.sess_limit
Current session limit.

mode_summary.act_sess_count
Total number of active sessions using the mode. If the **active_sessions** filter has been set to AP_YES, then this field will always be greater than zero.

mode_summary.fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

mode_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

mode_detail.mode_name
Mode name which designates the network properties for a group of sessions. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

mode_detail.description
Resource description (as specified on DEFINE_MODE).

mode_detail.sess_limit
Current session limit.

mode_detail.act_sess_count
Total number of active sessions using the mode. If the **active_sessions** filter has been set to AP_YES, then this field will always be greater than zero.

mode_detail.fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

mode_detail.min_conwinners_source
Specifies the minimum number of sessions on which the local LU is the contention winner (or 'first speaker').

mode_detail.min_conwinners_target
Specifies the minimum number of sessions on which the local LU is the contention loser (or 'bidder').

mode_detail.drain_source
Specifies whether the local LU satisfies waiting session requests before deactivating a session when session limits are changed or reset (AP_NO or AP_YES).

mode_detail.drain_partner
Specifies whether the partner LU satisfies waiting session requests before deactivating a session when session limits are changed or reset (AP_NO or AP_YES).

mode_detail.auto_act
Number of contention winner sessions that are automatically activated following the 'Change Number of Sessions' exchange with the partner LU.

mode_detail.act_cw_count
Number of active, contention winner (or 'first speaker') sessions using this mode. (The local LU does not need to 'bid' before using one of these sessions.)

mode_detail.act_cl_count
Number of active, contention loser (or 'bidder') sessions using this mode. (The local LU must 'bid' before using one of these sessions.)

mode_detail.sync_level
Specifies the synchronization levels supported by the mode (AP_NONE, AP_CONFIRM, or AP_SYNCPT).

mode_detail.default_ru_size
Specifies whether a default upper bound for the maximum RU size will be used. If this parameter has a value of AP_YES, the

mode_chars.max_ru_size_upp field specified on **define_mode** is ignored, and the upper bound for the maximum RU size is set to the link BTU size minus the size of the TH and the RH.

AP_YES
AP_NO

mode_detail.max_neg_sess_limit
Maximum negotiable session limit. Specifies the maximum session limit for the mode name that a local LU can use during its CNOS processing as the target LU.

mode_detail.max_rcv_ru_size
Maximum received RU size.

mode_detail.pending_session_count
Specifies the number of sessions pending (waiting for session activation to complete).

mode_detail.termination_count
If a previous CNOS verb has caused the mode session limit to be reset to zero, there may have been conversations using, or waiting to use these sessions. This field is a count of how many of these sessions have not yet been deactivated.

mode_detail.implicit
Specifies whether the entry was put in place because of an implicit (AP_YES) or explicit (AP_NO) definition.

Returned Parameters: Parameter Check

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

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

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MODE_DEFINITION

QUERY_MODE_DEFINITION returns both information previously passed in on a DEFINE_MODE verb and information about SNA-defined default modes.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific mode, or to obtain the list information in several "chunks," the **mode_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered by the **mode_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering).

If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

This verb returns definition information only. The QUERY_MODE verb returns information that is determined once the mode starts to be used by a local LU with a partner LU.

VCB Structure

```
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     reserv1[9];           /   reserved                           /
} MODE_CHARS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_MODE_DEFINITION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **mode_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

mode_name Mode name which designates the network properties for a group of sessions. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

mode_def_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

mode_def_summary.mode_name
8-byte mode name which designates the network properties for a group of sessions.

mode_def_summary.description
Resource description (as specified on DEFINE_MODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

mode_def_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

mode_def_detail.mode_name
Mode name which designates the network properties for a group of sessions. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

mode_def_detail.mode_chars.description
Resource description (as specified on DEFINE_MODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

mode_def_detail.mode_chars.max_ru_size_upp
Upper boundary for the maximum RU size to be used on sessions with this mode name.

mode_def_detail.mode_chars.receive_pacing_win
Specifies the session pacing window for the sessions when fixed pacing is used.

mode_def_detail.mode_chars.default_ru_size
Specifies whether a default upper bound for the maximum RU size will be used. If this parameter specifies AP_YES, **max_ru_size_upp** is ignored.

AP_YES
AP_NO

mode_def_detail.mode_chars.max_neg_sess_lim
Maximum negotiable session limit. Value used to negotiate the maximum number of sessions permissible between the local LU and the partner LU for the designated mode name.

mode_def_detail.mode_chars.plu_mode_session_limit
Session limit to negotiate initially on this mode. This value indicates a preferred session limit and is used for implicit CNOS.
Range: 0—32 767

mode_def_detail.mode_chars.min_conwin_src
Minimum number of contention winner sessions activatable by local LU using this mode.
Range: 0—32 767

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

mode_def_detail.mode_chars.cryptography
Reserved.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_MODE_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MODE_TO_COS_MAPPING

QUERY_MODE_TO_COS_MAPPING returns information about the mode to COS mapping.

The information is returned as a formatted list. To obtain information about a specific mode, or to obtain the list information in several “chunks,” the **mode_name** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

This list is ordered by the **mode_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with IBM's 6611 APPN MIB ordering). If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

If the default COS (which unknown modes are mapped to) has been overridden using DEFINE_MODE, QUERY_MODE_TO_COS_MAPPING also returns an entry with null **mode_name** (all zeros) and the default COS.

This entry is first in the ordering.

VCB Structure

```
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[2 ];             / reserved                     /
} MODE_TO_COS_MAPPING_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_MODE_TO_COS_MAPPING

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information: The **mode_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

mode_name Mode name which designates the network properties for a group of sessions. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST. This may be set to all zeros to indicate the entry for the default COS.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

mode_to_cos_mapping_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

mode_to_cos_mapping_data.mode_name
8-byte mode name which designates the network properties for a group of sessions. If this is set to all zeros, it indicates the entry for the default COS.

mode_to_cos_mapping_data.cos_name
Class of Service name associated with the mode name. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_MODE_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_NMVT_APPLICATION

QUERY_NMVT_APPLICATION returns a list of applications that have registered for network management vector transport (NMVT) level messages by previously issuing the REGISTER_NMVT_APPLICATION verb (see Chapter 14, “Management Services Verbs” for more details).

The information is returned as a list. To obtain information about a specific application, or to obtain the list information in several “chunks,” the **application** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

VCB Structure

```
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[2 ];             / reserved                            /
} NMVT_APPLICATION_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_NMVT_APPLICATION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information: The **application** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

application Application name. The name is an 8-byte alphanumeric type-A EBCDIC character string or all EBCDIC zeros. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Supplied Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
The number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

nmvt_application_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

nmvt_application_data.application
Name of registered application. The name is an 8-byte alphanumeric type-A EBCDIC character string.

nmvt_application_data.ms_vector_key_type
MS vector key accepted by the application. When the application registers for NMVT messages, it specifies which MS vector keys it will accept. For more information on NMVT application registration see Chapter 14, “Management Services Verbs.”

nmvt_application_data.conversion_required
Specifies whether the registered application requires messages to be converted from NMVT to MDS_MU format (AP_YES or AP_NO). When the application registers for NMVT messages, it will specify whether this conversion is required. For more information on NMVT application registration, see Chapter 14, “Management Services Verbs.”

Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_APPLICATION_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_NODE

QUERY_NODE returns node specific information and statistics. In addition to returning information determined dynamically during execution, QUERY_NODE also returns the parameters passed on the START_NODE verb.

VCB Structure

```
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;

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    reserv2[678];           /* reserved                         */
    unsigned char     node_spec_data_len;     /* length of node specific data     */
    unsigned char     ptf[64];                /* program temporary fix array      */
} CP_CREATE_PARMS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_NODE

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

cp_create_parms.crt_parms_len
Length of create parameters structure.

cp_create_parms.description
Resource description (as specified on START_NODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

cp_create_parms.node_type
It is always

AP_END_NODE

cp_create_parms.fqcp_name
Node's 17-byte fully qualified CP name. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name has a maximum length of 8 bytes with no embedded spaces.)

cp_create_parms.cp_alias
Locally used CP alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

cp_create_parms.mode_to_cos_map_supp
Specifies whether mode to COS mapping is supported by the node (AP_YES or AP_NO). If this is set to AP_YES then the COS specified on a DEFINE_MODE verb must either be an SNA defined COS or have been defined by issuing a DEFINE_COS verb.

cp_create_parms.mds_supported
Specifies whether Management Services supports Multiple Domain Support and MS Capabilities (AP_YES or AP_NO).

cp_create_parms.node_id
Node identifier used in XID exchange. This a 4-byte hexadecimal string.

cp_create_parms.max_locates
Maximum number of locates that the node can process.

cp_create_parms.dir_cache_size
Network Node only: Size of the directory cache.

cp_create_parms.max_dir_entries
Maximum number of directory entries. This is unlimited if this field is set to zero.

cp_create_parms.locate_timeout
Specifies the time in seconds before a network search will timeout. A value of zero indicates that the search has no timeout.

cp_create_parms.reg_with_nn
Specifies whether resources will be registered with the Network Node Server (AP_YES or AP_NO). If this field is set to AP_YES then the end node's network node server will only forward directed locates to it. If it

is not set, the network node server will forward all broadcast searches to the end node. Registration failure does not affect successful completion of the START_NODE verb. See “REGISTRATION_FAILURE” on page 336 for details.

cp_create_parms.reg_with_cds
Specifies whether the Network Node Server is allowed to register End Node resources with a Central Directory Server (AP_YES or AP_NO). (This field is ignored if **reg_with_nn** is set to AP_NO.) Network Node: Specifies whether local or domain resources can be optionally registered with Central Directory Server (AP_YES or AP_NO). Registration failure does not affect successful completion of the START_NODE verb.

cp_create_parms.mds_send_alert_q_size
Size of the MDS send alert queue. When this limit is reached, the MDS component deletes the oldest entry on the queue.

cp_create_parms.cos_cache_size
Size of the COS Database weights cache.

cp_create_parms.tree_cache_size
Size of the Topology Database routing tree cache size.

cp_create_parms.tree_cache_use_limit
Maximum number of uses of a cached tree. Once this number is exceeded, the tree is discarded and recomputed. This allows the node to balance sessions among equal weight routes. A low value provides better load balancing at the expense of increased activation latency.

cp_create_parms.max_tdm_nodes
Maximum number of nodes that can be stored in Topology Database (zero means unlimited).

cp_create_parms.max_tdm_tgs
Maximum number of TGs that can be stored in Topology Database (zero means unlimited).

cp_create_parms.max_isr_sessions
Maximum number of ISR sessions the node can participate in at once.

cp_create_parms.isr_sessions_upper_threshold
See **cp_create_parms.isr_sessions_lower_threshold**

cp_create_parms.isr_sessions_lower_threshold
The upper and lower thresholds control the node's congestion status. The node state changes from uncongested to congested if the number of ISR sessions exceeds the upper threshold. The node state changes back to uncongested once the number of ISR sessions dips below the lower threshold.

cp_create_parms.isr_max_ru_size
Maximum RU size supported for intermediate sessions.

cp_create_parms.isr_rcv_pac_window
Suggested receive pacing window size for intermediate sessions. This value is only used on the secondary hop of intermediate sessions if the adjacent node does not support adaptive pacing.

cp_create_parms.store_endpt_rscvs
Specifies whether RSCVs are stored for diagnostic purposes (AP_YES or AP_NO).

cp_create_parms.store_isr_rscvs
Specifies whether RSCVs are stored for diagnostic purposes (AP_YES or AP_NO).

cp_create_parms.store_dlur_rscvs
Specifies whether the node stores RSCVs for diagnostic purposes (AP_YES or AP_NO). If this field is set to AP_YES, then an RSCV is returned on the QUERY_DLUR_LU verb.

cp_create_parms.dlur_support
Specifies whether DLUR is supported (AP_YES or AP_NO).

cp_create_parms.pu_conc_support
Specifies whether PU concentration is supported (always AP_NO).

cp_create_parms.nn_rar
The Network Node's route additional resistance.

cp_create_parms.hpr_support
Specifies the level of support for HPR that is provided by the node (AP_NONE, AP_BASE, or AP_RTP).

cp_create_parms.mobile
Specifies whether the node is an HPR path-switch controller (AP_YES or AP_NO). If the **cp_create_parms.hpr_support** field is not set to AP_RTP this field is reserved.

cp_create_parms.discovery_support
Specifies whether Discovery functions are utilized by this node.

AP_DISCOVERY_CLIENT
Discovery client functions are used by this node

AP_DISCOVERY_SERVER
Discovery server functions are used by this node.

cp_create_parms.discovery_group_name
Specifies the group name used on Discovery functions utilized by the node. If this field is set to all zeros, the default group name is used.

cp_create_parms.node_spec_data_len
This field should always be set to zero.

cp_create_parms.ptf
Array for configuring and controlling future program temporary fix (ptf) operation.

cp_create_parms.ptf[0]
REQDISCONT support. Personal Communications normally uses REQDISCONT to deactivate limited resource host links that are no longer required by session traffic. This byte can be used to suppress Personal Communications' use of REQDISCONT, or to modify the settings used on REQDISCONT requests sent by Personal Communications.

AP_SUPPRESS_REQDISCONT
If this bit is set, Personal Communications does not use REQDISCONT (all other bits in this byte are ignored).

AP_OVERRIDE_REQDISCONT
If this bit is set, Personal Communications overrides the normal settings on REQDISCONT, based on the following two bits.

AP_REQDISCONT_TYPE
If this bit is set, Personal Communications specifies a type of 'immediate' on REQDISCONT. Otherwise, Personal Communications specifies a type of 'normal'. (This bit is ignored if AP_OVERRIDE_REQDISCONT is not set.)

AP_REQDISCONT_RECONTACT
If this bit is set, Personal Communications specifies 'immediate recontact' in REQDISCONT. Otherwise, Personal Communications specifies 'no immediate recontact'. (This bit is ignored if AP_OVERRIDE_REQDISCONT is not set.)

cp_create_parms.ptf[1]
ERP support.

Personal Communications normally processes an ACTPU(ERP) as an ERP (ACTPU(ERP) requests the PU-SSCP session be reset, but, unlike ACTPU(cold), does not request implicit deactivation of the subservient LU-SSCP and PLU-SLU sessions). SNA implementations may legally process ACTPU(ERP) as if it were ACTPU(cold).

AP_OVERRIDE_ERP
If this bit is set, Personal Communications processes all ACTPU requests as ACTPU(cold).

cp_create_parms.ptf[2]
BIS support.

Personal Communications normally uses the BIS protocol prior to deactivating a limited resource LU6.2 session. This byte allows the use of BIS to be overridden.

AP_SUPPRESS_BIS
If this bit is set, Personal Communications does not use the BIS protocol. Limited resource LU6.2 session are deactivated immediately using UNBIND(cleanup).

up_time Time (in hundredths of a second) since the node was started (or restarted).

mem_size
Size of the available storage, as obtained by storage management from the underlying operating system.

mem_used
Number of bytes of storage that are currently allocated to a process.

mem_warning_threshold
Allocation threshold beyond which storage management considers the storage resources to be constrained.

mem_critical_threshold
Allocation threshold beyond which storage management considers the storage resources to be critically constrained.

nn_functions_supported
Reserved.

functions_supported
Specifies which functions are supported. This may be one or more of the following:

AP_NEGOTIABLE_LS
AP_SEGMENT_REASSEMBLY
AP_BIND_REASSEMBLY
AP_PARALLEL_TGS
AP_CALL_IN
AP_ADAPTIVE_PACING

en_functions_supported
Specifies the End Node functions supported.

AP_SEGMENT_GENERATION
Node supports segment generation.

AP_MODE_TO_COS_MAP
Node supports mode name to COS name mapping.

AP_LOCATE_CDINIT
Node supports generation of locates and cross-domain initiate GDS variables for locating remote LUs.

AP_REG_WITH_NN
Node will register its LUs with the adjacent serving Network Node.

AP_REG_CHARS_WITH_NN
Node supports send register characteristics (can only be supported when send registered names is also supported).

nn_status
Reserved.

nn_frsn Reserved.

nn_rsn Reserved.

def_ls_good_xids
Total number of successful XID exchanges that have occurred on all defined link stations since the node was last started.

def_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all defined link stations since the node was last started.

dyn_ls_good_xids
Total number of successful XID exchanges that have occurred on all dynamic link stations since the node was last started.

dyn_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all dynamic link stations since the node was last started.

dlur_release_level
Specifies the current DLUR release level.

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_PARTNER_LU

QUERY_PARTNER_LU returns information about partner LUs that have been used by a local LU.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific partner LU, or to obtain the list information in several "chunks," the **plu_alias** field should be set (or the **fqplu_name** if the **plu_alias** is set to all zeros). If the **list_options** field is set to AP_FIRST_IN_LIST, both of these fields will be ignored. The **lu_name** or **lu_alias** field must always be set. The **lu_name**, if nonzero, will be used in preference to the **lu_alias**. See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered by the **fqplu_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering). If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

If **plu_alias** is set to all zeros, the **fqplu_name** value will be used; otherwise, the **plu_alias** is always used and the **fqplu_name** is ignored.

The list of partner LUs returned may be filtered according to whether they currently have any active sessions. If filtering is desired, the **active_sessions** field should be set to AP_YES (otherwise this field should be set to AP_NO).

This verb returns information that is determined when at least one session is established with the partner LU.

The QUERY_PARTNER_LU_DEFINITION verb returns definition information only.

VCB Structure

```
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    reserva[19];           /* reserved                          */
} PLU_DETAIL;
```

The application supplies the following parameters:

opcode AP_QUERY_PARTNER_LU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information.

The combination of the **lu_name** (or **lu_alias** if the **lu_name** is set to all zeros) and **plu_alias** (or **fqplu_name** if the

plu_alias is set to all zeros) specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned:

AP_FIRST_IN_LIST
The **plu_alias** and **fqplu_name** fields are ignored and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

lu_name LU name. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the index.

lu_alias Locally defined LU alias. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_name** and the **lu_alias** are set to all zeros then the LU associated with the CP (the default LU) is used.

plu_alias Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If this field is set to all zeros, the **fqplu_name** field will be used as the index value.

fqplu_name 17-byte fully qualified network name for the partner LU. This name is 17 bytes long and is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

active_sessions Active session filter. Specifies whether the returned partner LUs should be filtered according to whether they currently have any active sessions (AP_YES or AP_NO).

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

plu_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

plu_summary.plu_alias
Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

plu_summary.fqplu_name
17-byte fully qualified network name for the partner LU. This name is 17 bytes long and is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

plu_summary.description
Resource description (as specified on DEFINE_PARTNER_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

plu_summary.act_sess_count
Total number of active sessions between the local LU and the partner LU. If the **active_sessions** filter has been set to AP_YES, then this field will always be greater than zero.

plu_summary.partner_cp_name
17-byte fully qualified network name for the control point of the partner LU. This name is composed of two type A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

plu_summary.partner_lu_located
Specifies whether the CP name for the partner LU has been resolved (AP_YES or AP_NO).

plu_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

plu_detail.plu_alias
Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

plu_detail.fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

plu_detail.description
Resource description (as specified on DEFINE_PARTNER_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

plu_detail.act_sess_count
Total number of active sessions between the local LU and the partner LU. If the **active_sessions** filter has been set to AP_YES, then this field will always be greater than zero.

plu_detail.partner_cp_name
17-byte fully qualified network name for the control point of the partner LU. This name is composed of two type A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

plu_detail.partner_lu_located
Specifies whether the CP name for the partner LU has been resolved (AP_YES or AP_NO).

plu_detail.plu_un_name
Uninterpreted name of the partner LU. This is an 8-byte type-A EBCDIC character string.

plu_detail.parallel_sess_supp
Specifies whether parallel sessions are supported (AP_YES or AP_NO).

plu_detail.conv_security
Specifies whether conversation security information can be sent to this partner LU (AP_YES or AP_NO). If it is set to AP_NO, then any security information supplied by a TP is not sent to the partner LU.

plu_detail.max_mc_ll_send_size
Maximum size of logical length (LL) record that can be sent to the partner LU. Data records that are larger than this are broken down into several LL records before being sent to the partner LU. The maximum value **max_mc_ll_send_size** can take is 32767.

plu_detail.implict
Specifies whether the entry is the result of an implicit (AP_YES) or explicit (AP_NO) definition.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_PLU_NAME AP_INVALID_LU_NAME AP_INVALID_LU_ALIAS AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_PARTNER_LU_DEFINITION

QUERY_PARTNER_LU_DEFINITION returns information which had previously been passed in on a DEFINE_PARTNER_LU verb.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific partner LU, or to obtain the list information in several "chunks," the **plu_alias** field (or the **fqplu_name** if the **plu_alias** is set to all zeros) should be set. If the **plu_alias** field is nonzero it will be used to determine the index and the **fqplu_name** is ignored. If the **plu_alias** field is set to all zeros, the **fqplu_name** will be used to determine the index. If the **list_options** field is set to AP_FIRST_IN_LIST then both of these fields will be ignored. (In this case the returned list will be ordered by **plu_alias** if the AP_LIST_BY_ALIAS **list_options** is set, otherwise it will be ordered by **fqplu_name**). See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered on either **plu_alias** or **fqplu_name** according to the options specified. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering). If AP_LIST_FROM_NEXT is selected the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

Note this verb returns definition information only. The QUERY_PARTNER_LU verb returns information that is determined when at least one session is established with the partner LU.

VCB Structure

```
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    reserv1;               /  reserved                            /
   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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_PARTNER_LU_DEFINITION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **plu_alias** (or the **fqplu_name** if the **plu_alias** is set to all zeros) specified (see below) represents an index value that is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

AP_LIST_BY_ALIAS
The returned list is ordered by **plu_alias**. This option is only valid when AP_FIRST_IN_LIST is specified. If AP_LIST_FROM_NEXT or AP_LIST_INCLUSIVE is specified, the list ordering will depend on whether the **plu_alias** or **fqplu_name** has been supplied as a starting point.

plu_alias Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If this field is set to all zeros, the **fqplu_name** field is used to specify the required partner LU. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

fqplu_name 17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is only significant if the **plu_alias** field is set to all zeros. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**

partner_lu_def_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

partner_lu_def_summary.plu_alias
Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

partner_lu_def_summary.fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

partner_lu_def_summary.description
Resource description (as specified on DEFINE_PARTNER_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

partner_lu_def_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

partner_lu_def_detail.plu_alias
Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

partner_lu_def_detail.fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

partner_lu_def_detail.plu_chars.fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

partner_lu_def_detail.plu_chars.plu_alias
Partner LU alias.

partner_lu_def_detail.plu_chars.description
Resource description (as specified on DEFINE_PARTNER_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

partner_lu_def_detail.plu_chars.plu_un_name
Uninterpreted name of the partner LU. This is an 8-byte type-A EBCDIC character string.

partner_lu_def_detail.plu_chars.max_mc_ll_send_size
Maximum size of logical length (LL) record that can be sent to the partner LU. Data records that are larger than this are broken down into several LL records before being sent to the partner LU. The maximum value **max_mc_ll_send_size** can take is 32767.

partner_lu_def_detail.plu_chars.conv_security_ver
Specifies whether the partner LU is authorized to validate **user_ids** on behalf of local LUs, that is whether the partner LU may set the already verified indicator in an Attach request.

AP_YES
AP_NO

partner_lu_def_detail.plu_chars.parallel_sess_supp
Specifies whether parallel sessions are supported (AP_YES or AP_NO).

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_PLU_NAME
AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_PORT

QUERY_PORT returns a list of information about a node's ports. This information is structured as “determined data” (data gathered dynamically during execution) and “defined data” (the data supplied by the application on DEFINE_PORT).

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific port, or to obtain the list information in several “chunks,” the **port_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See “Querying the Node” on page 12, for background on how the list formats are used.

This list is ordered by the **port_name**. Ordering is by name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with IBM's 6611 APPN MIB ordering). If AP_LIST_FROM_NEXT is selected, the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

The list of ports returned may be filtered by the name of the DLC that they belong to. In this case the **dlc_name** field should be set (otherwise this field should be set to all zeros).

VCB Structure

```
typedef struct query_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    port_name[8];        /* port name                    */
    unsigned char    dlc_name[8];         /* dlc name filter              */
} QUERY_PORT;

typedef struct port_summary
{
    unsigned short   overlay_size;        /* size of this entry           */
    unsigned char    port_name[8];        /* port name                    */
    unsigned char    description[RD_LEN];
description     /
    unsigned char    port_state;          /* port state                   */
    unsigned char    reserv1[1];          /* reserved                     */
    unsigned char    dlc_name[8];         /* name of DLC                  */
} PORT_SUMMARY;

typedef struct port_detail
{
    unsigned short   overlay_size;        /* size of this entry           */
    unsigned char    port_name[8];        /* port name                    */
```

```
    unsigned  char    reserv1[2];          /  reserved                          /
    PORT_DET_DATA     det_data;              /  determined  data                  /
    PORT_DEF_DATA     def_data;              /  defined  data                     /
} PORT_DETAIL;

typedef struct port_det_data
{
     unsigned char     port_state;          /  port state                        /
     unsigned char     dlc_type;            /  DLC type                         /
     unsigned char     port_sim_rim;        /  port initialization options      /
    unsigned  char     reserv1;             /  reserved                         /
     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           /
                                            /  LS count                         /
    unsigned  char     reserva[2 ];         /  reserved                         /
} PORT_DET_DATA;

} PORT_DET_DATA;
typedef struct port_def_data
{
    unsigned  char     description[RD_LEN];
                                            /  resource description             /
     unsigned char     dlc_name[8];         /  DLC name associated with port    /
     unsigned char     port_type;           /  port type                        /
     unsigned char     reserv3[7];          /  unsigned char                    /
     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 lim    /
     unsigned char     ls_role;             /  initial link station role       /
    unsigned  char     reserv1[27];         /  reserved                         /
    unsigned  short    implicit_deact_timer;
                                            /  Implicit link HPR link           /
                                            /  deactivation timer               /
    unsigned  short    act_xid_exchange_limit;
                                            /  activation XID exchange lim      /
    unsigned  short    nonact_xid_exchange_limit;
                                            /  non-act. XID exchange lim        /
     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&lbraket.SIZEOF_PORT_SPEC_DATA];

}   PORT_SPEC_DATA;

typedef struct link_spec_data
{
         unsigned  char   link_data&lbraket.SIZEOF_LINK_SPEC_DATA];

}   LINK_SPEC_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_PORT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information.

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **port_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

port_name Name of port being queried. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

dlc_name DLC name filter. This should be set to all zeros or an 8-byte string in a locally displayable character set. If this field is set then only ports belonging to this DLC are returned. This field is ignored if it is set to all zeros.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

port_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

port_summary.port_name
Name of port associated with this link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

port_summary.description
Resource description (as specified on DEFINE_PORT). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

port_summary.port_state
Specifies the current state of the port.

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

port_summary.dlc_name
Name of the DLC. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

port_detail.overlay_size
The number of bytes in this entry (including any **link_spec_data**), and hence the offset to the next entry returned (if any).

port_detail.port_name
Name of port associated with this link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

port_detail.det_data.port_state
Specifies the current state of the port.

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

port_detail.det_data.dlc_type
Type of DLC. Personal Communications supports the following types:

AP_LLC2
AP_SDLC
AP_TWINAX

port_detail.det_data.port_sim_rim
Specifies whether Set Initialization Mode (SIM) and Receive Initialization Mode (RIM) are supported (AP_YES or AP_NO).

port_detail.det_data.def_ls_good_xids
Total number of successful XID exchanges that have occurred on all defined link stations on this port since the last time this port was started.

port_detail.det_data.def_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all defined link stations on this port since the last time this port was started.

port_detail.det_data.dyn_ls_good_xids
Total number of successful XID exchanges that have occurred on all dynamic link stations on this port since the last time this port was started.

port_detail.det_data.dyn_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all dynamic link stations on this port since the last time this port was started.

port_detail.def_data.description
Resource description (as specified on DEFINE_PORT). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

port_detail.def_data.dlc_name
Name of associated DLC. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

port_detail.def_data.port_type
Specifies the type of line used by the port. The value corresponds to one of the following:

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

port_detail.def_data.port_number
Port number.

port_detail.def_data.max_rcv_btu_size
Maximum BTU size that can be received.

port_detail.def_data.tot_link_act_lim
Total link activation limit.

port_detail.def_data.inb_link_act_lim
Inbound link activation limit.

port_detail.def_data.out_link_act_lim
Outbound link activation limit.

port_detail.def_data.ls_role
Link station role. This can be negotiable (AP_LS_NEG), primary (AP_LS_PRI), or secondary (AP_LS_SEC). reserved if **implicit_hpr_support** is set to AP_NO.

port_detail.def_data.implicit_deact_timer
Limited resource link deactivation timer. A limited resource HPR-capable implicit link is automatically deactivated if no data traverses the link for the duration of this timer, and no sessions are using the link. Note that this parameter is reserved unless **implicit_limited_resource** and **implicit_hpr_support** are both set to AP_YES.

port_detail.def_data.act_xid_exchange_limit
Activation XID exchange limit.

port_detail.def_data.nonact_xid_exchange_limit
Non-activation XID exchange limit.

port_detail.def_data.ls_xmit_rcv_cap
Specifies the link station transmit/receive capability. This is either two-way simultaneous (AP_LS_TWS) or two way alternating (AP_LS_TWA).

port_detail.def_data.max_ifrm_rcvd
Maximum number of I-frames that can be received by local link stations before an acknowledgement is sent. Range: 1—127

port_detail.def_data.target_pacing_count
Numeric value between 1 and 32767 inclusive indicating the desired pacing window size for BINDs on this TG. The number is only significant when fixed bind pacing is being performed. Personal Communications does not currently use this value.

port_detail.def_data.max_send_btu_size
Maximum BTU size that can be sent.

port_detail.def_data.dlc_data.length
Port address length.

port_detail.def_data.dlc_data.address
Port address.

port_detail.def_data.hpr_dlc_data.length
HPR Port address length.

port_detail.def_data.hpr_dlc_data.address
HPR Port address. This is currently used when supporting HPR links. The field specifies the information sent by Personal Communications in the X'80' subfield of the X'61' control vector on XID3 exchanged on link stations using this port.

port_detail.def_data.implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are permitted for implicit link stations off this port (AP_YES or AP_NO).

port_detail.def_data.implicit_limited_resource
Specifies whether implicit link stations off this port are defined as limited resources (AP_YES or AP_NO).

port_detail.def_data.implicit_hpr_support
Specifies whether HPR is supported on implicit links (AP_YES or AP_NO).

port_detail.def_data.implicit_link_lvl_error
Specifies whether HPR traffic is sent on implicit links using link-level error recovery (AP_YES or AP_NO). Note that the parameter is

port_detail.def_data.default_tg_chars
TG characteristics (See “DEFINE_COS” on page 35). These are used for implicit link stations off this port and also for defined link stations which specify **use_default_tg_chars**.

port_detail.def_data.discovery_supported
Specifies whether Discovery search functions are performed on this port (AP_YES or AP_NO).

port_detail.def_data.port_spec_data_len
Unpadded length, in bytes, of data passed unchanged to port on ACTIVATE_PORT signal. The data is concatenated to the PORT_DETAIL structure.

port_detail.def_data.link_spec_data_len
Data passed unchanged to link station component during initialization. The data is concatenated to the PORT_DETAIL structure immediately following the port specific data. The port specific data and the link specific data together will be padded to end on a 4-byte boundary. There will be no explicit padding between the port specific data and the link specific data.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_PORT_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_PU

QUERY_PU returns a list of local PUs and the links associated with them.

The information is returned as a list. To obtain information about a specific PU, or to obtain the list information in several "chunks," the **pu_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See "Querying the Node" on page 12, for background on how the list formats are used.

The verb specifies whether local PUs are attached directly to the host system or attached via DLUR. The **host_attachment** field can be used as a filter so that only information about the specified attachment type is returned.

VCB Structure

```
typedef struct query_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];              / 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   reserva[2 ];             / 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    reserve;               /  reserved                         /
} SESSION_STATS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_PU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information.

The **pu_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

pu_name Name of the first PU to be listed. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. PU name. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

host_attachment Filter for host attachment:

AP_NONE
Return information about all local PUs.

AP_DLUR_ATTACHED
Return information about all local PUs which are supported by DLUR.

AP_DIRECT_ATTACHED
Return information about only those PUs which are directly attached to the host system.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

pu_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

pu_data.pu_name
PU Name. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

pu_data.description
Resource description (as specified on DEFINE_LS or DEFINE_INTERNAL_PU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

pu_data.ls_name
Name of the link station associated with this PU. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

pu_data.pu_sscp_sess_active
Specifies whether the PU-SSCP session is active (AP_YES or AP_NO).

pu_data.host_attachment
Local PU host attachment type:

AP_DLUR_ATTACHED
PU is attached to host system using DLUR.

AP_DIRECT_ATTACHED
PU is directly attached to host system.

pu_data.pu_sscp_stats.rcv_ru_size
This field is always reserved.

pu_data.pu_sscp_stats.send_ru_size
This field is always reserved.

pu_data.pu_sscp_stats.max_send_btu_size
Maximum BTU size that can be sent.

pu_data.pu_sscp_stats.max_rcv_btu_size
Maximum BTU size that can be received.

pu_data.pu_sscp_stats.max_send_pac_win
This field will always be set to zero.

pu_data.pu_sscp_stats.cur_send_pac_win
This field will always be set to zero.

pu_data.pu_sscp_stats.max_rcv_pac_win
This field will always be set to zero.

pu_data.pu_sscp_stats.cur_rcv_pac_win
This field will always be set to zero.

pu_data.pu_sscp_stats.send_data_frames
Number of normal flow data frames sent.

pu_data.pu_sscp_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

pu_data.pu_sscp_stats.send_data_bytes
Number of normal flow data bytes sent.

pu_data.pu_sscp_stats.rcv_data_frames
Number of normal flow data frames received.

pu_data.pu_sscp_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

pu_data.pu_sscp_stats.rcv_data_bytes
Number of normal flow data bytes received.

pu_data.pu_sscp_stats.sidh
Session ID high byte.

pu_data.pu_sscp_stats.sidl
Session ID low byte.

pu_data.pu_sscp_stats.odai
Origin Destination Address Indicator. When bringing up a session, the sender of the ACTPU sets this field to zero if the local node contains the primary link station, and sets it to one if the ACTPU sender is the node containing the secondary link station.

pu_data.pu_sscp_stats.ls_name
Link station name associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_PU_NAME AP_INVALID_PU_TYPE AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_RTP_CONNECTION

QUERY_RTP_CONNECTION is used at a Network Node or an End Node and returns list information about Rapid Transport Protocol (RTP) connections for which the node is an endpoint.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific RTP connection, or to obtain the list information in several "chunks," the **rtp_name** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered by the **rtp_name**. Ordering is according to name length first, and then by ASCII lexicographical ordering for names of the same length (in accordance with normal MIB ordering). If AP_LIST_FROM_NEXT is selected the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

VCB Structure

```
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     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[4 ];             /   reserved                                /
} RTP_STATISTICS;
```

Note: The **rtp_connection_detail** overlay will be followed by a Route Selection Control Vector (RSCV) as defined by SNA Formats. This control vector defines the connection route through the network and is carried on the BIND.

Supplied parameters

The application supplies the following parameters:

opcode AP_QUERY_RTP_CONNECTION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information.

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **rtp_name** represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The **rtp_name** is ignored and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

rtp_name RTP connection name. This name is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

rtp_connection_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

rtp_connection_summary.rtp_name
RTP connection name. This name is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

rtp_connection_summary.first_hop_ls_name
Link station name of the first hop of the RTP connection. This name is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

rtp_connection_summary.dest_node_name
Fully-qualified, 17-byte name of the destination node of the RTP connection composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

rtp_connection_summary.cos_name
Class-of-service name for the RTP connection. This is an 8-byte alphanumeric type-A EBCDIC character string, padded to the right with EBCDIC spaces.

rtp_connection_summary.num_sess_active
Number of sessions currently active on the RTP connection.

rtp_connection_detail.overlay_size
The number of bytes in this entry (including any appended RSCV), and hence the offset to the next entry returned (if any).

rtp_connection_detail.rtp_name
RTP connection name. This name is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

rtp_connection_detail.first_hop_ls_name
Link station name of the first hop of the RTP connection. This name is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

rtp_connection_detail.dest_node_name
Fully-qualified, 17-byte name of the destination node of the RTP connection composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded-with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

rtp_connection_detail.cos_name
Class-of-service name for the RTP connection. This is an 8-byte alphanumeric type-A EBCDIC character string, padded to the right with EBCDIC spaces.

rtp_connection_detail.max_btu_size
Maximum btu size for the RTP connection measured in bytes.

rtp_connection_detail.liveness_timer
Liveness timer for the RTP connection, measured in seconds.

rtp_connection_detail.local_tcid
Local TCID for the RTP connection.

rtp_connection_detail.remote_tcid
Remote TCID for the RTP connection.

rtp_connection_detail.rtp_stats.bytes_sent
Total number of bytes that the local node has sent on this RTP connection.

rtp_connection_detail.rtp_stats.bytes_received
Total number of bytes that the local node has received on this RTP connection.

rtp_connection_detail.rtp_stats.bytes_resent
Total number of bytes resent by the local node owing to loss in transit.

rtp_connection_detail.rtp_stats.bytes_discarded
Total number of bytes sent by the other end of the RTP connection and were discarded as duplicates of data already received.

rtp_connection_detail.rtp_stats.packets_sent
Total number of packets that the local node has sent on this RTP connection.

rtp_connection_detail.rtp_stats.packets_received
Total number of packets that the local node has received on this RTP connection.

rtp_connection_detail.rtp_stats.packets_resent
Total number of packets resent by the local node owing to loss in transit.

rtp_connection_detail.rtp_stats.packets_discarded
Total number of packets sent by the other end of the RTP connection which were discarded as duplicates of data already received.

rtp_connection_detail.rtp_stats.gaps_detected
Total number of gaps detected by the local node. Each gap corresponds to one or more lost frames.

rtp_connection_detail.rtp_stats.send_rate
Current send rate on this RTP connection (measured in kilobits per second). This is the maximum allowed send rate as calculated by the ARB algorithm.

rtp_connection_detail.rtp_stats.max_send_rate
Maximum send rate on this RTP connection (measured in kilobits per second).

rtp_connection_detail.rtp_stats.min_send_rate
Minimum send rate on this RTP connection (measured in kilobits per second).

rtp_connection_detail.rtp_stats.receive_rate
Current receive rate on this RTP connection (measured in kilobits per second). This is the actual receive rate calculated over the last measurement interval.

rtp_connection_detail.rtp_stats.max_receive_rate
Maximum receive rate on this RTP connection (measured in kilobits per second).

rtp_connection_detail.rtp_stats.min_receive_rate
Minimum receive rate on this RTP connection (measured in kilobits per second).

rtp_connection_detail.rtp_stats.burst_size
Current burst size on the RTP Connection measured in bytes.

rtp_connection_detail.rtp_stats.up_time
Total number of seconds the RTP connection has been active.

rtp_connection_detail.rtp_stats.smooth_rtt
Smoothed measure of round-trip time between the local node and the partner RTP node (measured in milliseconds).

rtp_connection_detail.rtp_stats.last_rtt
The last measured round-trip time between the local node and the partner RTP node (measured in milliseconds).

rtp_connection_detail.rtp_stats.short_req_timer
The current duration used for the SHORT_REQ timer (measured in milliseconds).

rtp_connection_detail.rtp_stats.short_req_timeouts
Total number of times the SHORT_REQ timer has expired for this RTP connection.

rtp_connection_detail.rtp_stats.liveness_timeouts
Total number of times the liveness timer has expired for this RTP connection. The liveness timer expires when the connection has been idle for the period specified in **rtp_connection_detail.liveness_timer**.

rtp_connection_detail.rtp_stats.in_invalid_sna_frames
Total number of SNA frames received and discarded as invalid on this RTP connection.

rtp_connection_detail.rtp_stats.in_sc_frames
Total number of session control frames received on this RTP connection.

rtp_connection_detail.rtp_stats.out_sc_frames
Total number of session control frames sent on this RTP connection.

rtp_connection_detail.num_sess_active
Number of sessions currently active on the RTP connection.

rtp_connection_detail.rscv_len
Length of the appended Route Selection Control Vector for the RTP connection. (If none is appended, then the length is zero.) The RSCV will be padded to end on a 4-byte boundary to ensure correct alignment of the next detail entry, but the **rscv_len** does not include this padding.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_RTP_CONNECTION AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_SESSION

QUERY_SESSION returns list information about sessions for which the node is an end point.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific session, or to obtain the list information in several "chunks," the **session_id** field should be set. Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. Note that the **lu_name** (or **lu_alias**) and **plu_alias** (or **fqplu_name**) fields must always be set. The **lu_name**, if nonzero, will be used in preference to the **lu_alias**. See "Querying the Node" on page 12, for background on how the list formats are used.

If **plu_alias** is set to all zeros, the **fqplu_name** value will be used, otherwise the **plu_alias** is always used and the **fqplu_name** is ignored.

The list of sessions returned may be filtered by the name of the mode that they are associated with. In this case the **mode_name** field should be set (otherwise this field should be set to all zeros).

In addition to the detail information for each session, a Route Selection Control Vector (RSCV) will be returned (in key-length format) if this is specified on the START_NODE parameters. This RSCV (specified in SNA Formats) defines the route through the network which the session will take in a hop-by-hop form.

VCB Structure

```
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    reserv3a[3];         /  reserved                          /
    unsigned  char    reserva[2 ];         /  reserved                          /
    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    reserve;              /  reserved                     /
} SESSION_STATS;
```

Note: The session detail overlay may be followed by a Route Selection Control Vector (RSCV) as defined by SNA Formats. This control vector defines the session route through the network and is carried on the BIND. The RSCV is included (in key-length format) if the **store_endpt_rscvs** field on the START_NODE verb is set to AP_YES. If the START_NODE **store_endpt_rscvs** field is set to AP_NO then the **rscv_len** is set to zero.

Supplied parameters

The application supplies the following parameters:

opcode AP_QUERY_SESSION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information.

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The combination of **lu_name** (or **lu_alias** if the **lu_name** is set to all zeros), **pu_alias** (or **fqplu_name** if the **plu_alias** is set to all zeros), **mode_name** and **session_id** specified (see below) represent an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The **session_id** is ignored and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

lu_name LU name. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the index.

lu_alias Locally defined LU alias. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_name** and the **lu_alias** fields are set to all zeros, the LU associated with the CP (the default LU) is used.

plu_alias Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If this field is set to all zeros, the **fqplu_name** field will be used for determining the index.

fqplu_name 17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

mode_name Mode name filter. This should be set to all zeros or an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces. If this field is set then only sessions associated with this mode are returned. This field is ignored if it is set to all zeros.

session_id 8-byte identifier of the session. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

session_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

session_summary.plu_alias
Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

session_summary.fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces.

session_summary.mode_name
Mode name. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

session_summary.session_id
8-byte identifier of the session.

session_summary.fqpcid.pcid
Procedure Correlator ID. This is an 8-byte hexadecimal string.

session_summary.fqpcid.fqcp_name
Fully qualified CP name. This 17-byte name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

session_detail.overlay_size
The number of bytes in this entry (including any appended RSCV), and hence the offset to the next entry returned (if any).

session_detail.plu_alias
Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

session_detail.fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces.

session_detail.mode_name
Mode name. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

session_detail.session_id
8-byte identifier of the session.

session_detail.fqpcid.pcid
Procedure Correlator ID. This is an 8-byte hexadecimal string.

session_detail.fqpcid.fqcp_name
Fully qualified CP name. This 17-byte name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

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

session_detail.trans_pri
Transmission priority. This is set to one of the following:

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

session_detail.ltd_res
Specifies whether the session uses a limited resource link (AP_YES or AP_NO).

session_detail.polarity
Specifies the polarity of the session (AP_PRIMARY or AP_SECONDARY).

session_detail.contention
Specifies the session contention polarity. This indicates whether the local LU has 'first refusal' for the use of this session (AP_CONWINNER) or whether it must bid before using the session (AP_CONLOSER).

session_detail.sess_stats.rcv_ru_size
Maximum receive RU size.

session_detail.sess_stats.send_ru_size
Maximum send RU size.

session_detail.sess_stats.max_send_btu_size
Maximum BTU size that can be sent.

session_detail.sess_stats.max_rcv_btu_size
Maximum BTU size that can be received.

session_detail.sess_stats.max_send_pac_win
Maximum size of the send pacing window on this session.

session_detail.sess_stats.cur_send_pac_win
Current size of the send pacing window on this session.

session_detail.sess_stats.max_rcv_pac_win
Maximum size of the receive pacing window on this session.

session_detail.sess_stats.cur_rcv_pac_win
Current size of the receive pacing window on this session.

session_detail.sess_stats.send_data_frames
Number of normal flow data frames sent.

session_detail.sess_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

session_detail.sess_stats.send_data_bytes
Number of normal flow data bytes sent.

session_detail.sess_stats.rcv_data_frames
Number of normal flow data frames received.

session_detail.sess_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

session_detail.sess_stats.rcv_data_bytes
Number of normal flow data bytes received.

session_detail.sess_stats.sidh
Session ID high byte.

session_detail.sess_stats.sidl
Session ID low byte.

session_detail.sess_stats.odai
Origin Destination Address Indicator. When bringing up a session, the sender of the BIND sets this field to zero if the local node contains the primary link station. It sets it to one if the BIND sender is the node containing the secondary link station.

session_detail.sess_stats.ls_name
Link station name associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant. This field can be used to correlate the session statistics with the link over which session data flows.

session_detail.rscv_len
Length of the RSCV which is appended to the **session_detail** structure. (If none is appended, then the length is zero.) The RSCV will be padded to end on a 4-byte boundary.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_SESSION_ID AP_INVALID_LU_NAME AP_INVALID_LU_ALIAS AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

QUERY_STATISTICS

QUERY_STATISTICS queries link station and port statistics. Personal Communications passes this query directly to the DLC. The format of the statistics depends on the DLC implementation.

VCB Structure

```
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[2 ];         / reserved                     /
} QUERY_STATISTICS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_STATISTICS

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

name Name defined for the link station or port (depending on setting of **stats_type** parameter). This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. Personal Communications uses this to correlate the response to the correct link station or port.

stats_type The type of resource for which statistics are requested. This must be set to one of the following:

AP_LS
AP_PORT

table_type The type of statistics table requested. This must be set to one of the following categories of information.

AP_STATS_TBL
Specifies that statistical information will be returned.

AP_ADMIN_TBL
Specifies that administrative information will be returned.

AP_OPER_TBL
Specifies that operational information will be returned. The format of the information returned for each category is DLC implementation specific.

reset_stats Specifies whether the statistics should be reset (AP_YES or AP_NO).

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

dlc_type Type of the DLC. The value of this field is DLC implementation specific. The values are the following:

AP_LLC2
AP_SDLC
AP_TWINAX

statistics Current statistics of link station or port.

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_LINK_NAME
AP_INVALID_PORT_NAME
AP_INVALID_STATS_TYPE
AP_INVALID_TABLE_TYPE

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

primary_rc AP_STATE_CHECK

secondary_rc AP_LINK_DEACTIVATED
AP_PORT_DEACTIVATED

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_TP

QUERY_TP returns information about TPs currently being used by a local LU.

The information is returned as a list. To obtain information about a specific TP, or to obtain the list information in several "chunks," the **tp_name** field should be set. If the **list_options** field is set to AP_FIRST_IN_LIST then this field will be ignored. Note that the **lu_name** or **lu_alias** field must always be set. The **lu_name** field, if nonzero, will be used in preference to the **lu_alias** field. See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered by the **tp_name** using EBCDIC lexicographical ordering for names of the same length. This verb returns information that is determined once the TP starts to be used by a local LU. The QUERY_TP_DEFINITION verb returns definition information only.

VCB Structure

```
typedef struct query_tp
{
    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 char     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[2 ];           /   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 reserved[6];          /  max size is 12  bytes          /
} TP_SPEC_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_TP

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information: The combination of **lu_name** (or **lu_alias** if the **lu_name** is set to all zeros) and **tp_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned.

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

lu_name LU name. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the index.

lu_alias Locally defined LU alias. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_name** and the **lu_alias** are set to all zeros, the LU that is associated with the CP (the default LU) is used.

tp_name TP name. This is a 64-byte string, padded to the right with spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc	AP_OK
buf_size	Length of the information returned in the buffer.
total_buf_size	Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.
num_entries	Number of entries actually returned.
total_num_entries	Total number of entries that could have been returned. This may be higher than **num_entries**.
tp_data.overlay_size	The number of bytes in this entry, and hence the offset to the next entry returned (if any).
tp_data.tp_name	TP name. This is a 64-byte string, padded to the right with spaces.
tp_data.instance.description	Resource description (as specified on DEFINE_TP). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.
tp_data.instance_limit	Maximum number of concurrently active instances of the specified TP.
tp_data.instance_count	Number of instances of the specified TP that are currently active.
tp_data.locally_started_count	Number of instances of the specified TP which have been started locally (by the TP issuing a TP_STARTED verb).
tp_data.remotely_started_count	Number of instances of the specified TP that have been started remotely (by a received Attach request).
tp_chars.tp_data.pathname	Specifies the path and TP name.
tp_chars.tp_data.parameters	Specifies the parameters for the TP.
tp_chars.tp_data.queued	Specifies whether the TP will be queued.
tp_chars.tp_data.load_type	Specifies how the TP will be loaded.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_TP_NAME AP_INVALID_LU_NAME AP_INVALID_LU_ALIAS AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_TP_DEFINITION

QUERY_TP_DEFINITION returns both information previously passed in on a DEFINE_TP verb and information about Personal Communications defined TPs.

The information is returned as a list in one of two formats, either summary or detailed information. To obtain information about a specific TP, or to obtain the list information in several "chunks," the **tp_name** field should be set.

Otherwise (if the **list_options** field is set to AP_FIRST_IN_LIST), this field will be ignored. See "Querying the Node" on page 12, for background on how the list formats are used.

This list is ordered by the **tp_name**, using EBCDIC lexicographical ordering. If AP_LIST_FROM_NEXT is selected the returned list starts from the next entry according to the defined ordering (whether the specified entry exists or not).

This verb returns definition information only. The QUERY_TP verb returns information that is determined once the TP starts to be used by a local LU.

VCB Structure

```
typedef struct query_tp_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    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    reserv3[1 ];             /  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 reserved[6];                / max size is 12  bytes               /
} TP_SPEC_DATA;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_TP_DEFINITION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written. The application can append data to the end of the VCB in which case **buf_ptr** must be set to NULL.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information:

AP_SUMMARY
Returns summary information only.

AP_DETAIL
Returns detailed information. The **tp_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned:

AP_FIRST_IN_LIST
The index value is ignored, and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

tp_name Name of the defined TP. This is a 64-byte string, padded to the right with spaces. This field is ignored if **list_options** is set to AP_FIRST_IN_LIST.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

tp_def_summary.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

tp_def_summary.tp_name
Defined TP name. This is a 64-byte string, padded to the right with spaces.

tp_def_summary.description
Resource description (as specified on DEFINE_TP). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

tp_def_detail.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

tp_def_detail.tp_name
Defined TP name. This is a 64-byte string, padded to the right with spaces.

tp_def_detail.tp_chars.description
Resource description (as specified on DEFINE_TP). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

tp_def_detail.tp_chars.conv_type
Specifies the types of conversation supported by the TP:

AP_BASIC
AP_MAPPED
AP_EITHER

tp_def_detail.tp_chars.security_rqd
Specifies whether conversation security information is required to start the TP (AP_NONE, AP_SAME or AP_PGM).

tp_def_detail.tp_chars.sync_level
Specifies the synchronization levels supported by the TP:

AP_NONE
The TP supports a synchronization level of None.

AP_CONFIRM_SYNC_LEVEL
The TP supports a synchronization level of Confirm.

AP_EITHER
The TP supports a synchronization level of None or Confirm.

tp_def_detail.tp_chars.dynamic_load
Specifies whether the TP can be dynamically loaded (AP_YES or AP_NO).

tp_def_detail.tp_chars.enabled
Specifies whether the TP can be attached successfully (AP_YES or AP_NO).

tp_def_detail.tp_chars.pip_allowed
Specifies whether the TP can receive program initialization (PIP) parameters (AP_YES or AP_NO).

tp_def_detail.tp_chars.tp_instance_limit
Limit on the number of concurrently active TP instances.

tp_def_detail.tp_chars.incoming_alloc_timeout
Specifies the number of seconds that an incoming Attach will be queued waiting for a RECEIVE_ALLOCATE. Zero implies no timeout, and so it will be held indefinitely.

tp_def_detail.tp_chars.rcv_alloc_timeout
Specifies the number of seconds that a RECEIVE_ALLOCATE verb will be queued while waiting for an Attach. Zero implies no timeout, and so it will be held indefinitely.

tp_def_detail.tp_chars.tp_data_len
Length of the implementation-dependent TP data.

tp_def_detail.tp_chars.tp_data
Implementation-dependent TP data that is passed unchanged on the DYNAMIC_LOAD_INDICATION.

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_TP_NAME AP_INVALID_LIST_OPTION

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

Chapter 7. Session Limit Verbs

This chapter describes verbs used to initialize, change, or reset session limits.

© Copyright IBM Corp. 1989, 1996

CHANGE_SESSION_LIMIT

The CHANGE_SESSION_LIMIT verb requests that the session limits of a particular mode (or session group) be changed. Sessions may be activated or deactivated as a result of processing this verb.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode — AP_CHANGE_SESSION_LIMIT

format — Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name — LU name of the local LU requested to change session limits. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias — Alias of the local LU requested to change session limits. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must

be set. If both the **lu_name** and the **lu_alias** fields are set to all zeros then the verb is forwarded to the LU associated with the CP (the default LU).

plu_alias
Alias by which the partner LU is known to the local LU. This name must match the name of a partner LU established during configuration. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If this field is set to all zeros, the **fqplu_name** field is used to specify the required partner LU.

fqplu_name
Fully qualified LU name for the partner LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is only significant if the **plu_alias** field is set to all zeros.

mode_name
Name of a set of networking characteristics defined during configuration. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

set_negotiable
Specifies whether the maximum negotiable session limit for this mode should be modified to become the **plu_mode_session_limit**.

AP_YES
AP_NO

plu_mode_session_limit
Requested total session limit for this mode. The actual session limit (which may be negotiated with the partner LU), is the agreed maximum number of sessions supported between the local LU and the partner LU on this mode.

min_conwinners_source
Minimum number of sessions in this mode for which the local LU is the contention winner.

min_conwinners_target
Minimum number of sessions in this mode for which the partner LU is the contention winner.

auto_act
Number of sessions to automatically activate after the session limit is changed. The actual number of automatically activated sessions is the minimum of this value and the negotiated minimum number of contention winner sessions for the local LU. When sessions are deactivated normally (specifying AP_DEACT_NORMAL) below this limit, new sessions are activated up to this limit.

responsible Indicates whether the source (local) or target (partner) LU is responsible for deactivating sessions after the session limit is changed (AP_SOURCE or AP_TARGET).

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

secondary_rc AP_AS_SPECIFIED
AP_AS_NEGOTIATED

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc 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

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_MODE_RESET

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: Allocation Error

If the verb does not execute because of an allocation error, Personal Communications returns the following parameters:

primary_rc AP_ALLOCATION_ERROR

secondary_rc AP_ALLOCATION_FAILURE_NO_RETRY

sense_data Sense data associated with allocation error.

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters: primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

Returned Parameters: Other Errors

If the verb does not execute because of an error, Personal Communications returns the following parameters:

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

The INITIALIZE_SESSION_LIMIT verb initializes the mode session limits.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_INITIALIZE_SESSION_LIMIT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name LU name of the local LU requested to initialize session limits. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias Alias of the local LU requested to initialize session limits. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_name** and **lu_alias** are set to all zeros, the verb is forwarded to the LU associated with the CP (the default LU).

plu_alias Alias by which the partner LU is known to the local LU. This name must match the name of a partner LU established during configuration. This is an 8-byte string in a locally displayable character set. All 8 bytes are

significant and must be set. If this field is set to all zeros, the **fqplu_name** field is used to specify the required partner LU.

fqplu_name
Fully qualified LU name for the partner LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is only significant if the **plu_alias** field is set to all zeros.

mode_name
Name of a set of networking characteristics defined during configuration. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

set_negotiable
Specifies whether the maximum negotiable session limit for this mode should be modified to become the **plu_mode_session_limit**.

AP_YES
AP_NO

plu_mode_session_limit
Requested total session limit for this mode. The actual session limit (which may be negotiated with the partner LU), is the agreed maximum number of sessions supported between the local LU and the partner LU on this mode. This must be set to a value in the range one to 32767.

min_conwinners_source
Minimum number of sessions in this mode for which the local LU is the contention winner. This must be set to a value in the range zero to 32767.

min_conwinners_target
Minimum number of sessions in this mode for which the partner LU is the contention winner. This must be set to a value in the range zero to 32767.

auto_act Number of sessions to automatically activate after the session limit is changed. The actual number of automatically activated sessions is the minimum of this value and the negotiated minimum number of contention winner sessions for the local LU. When sessions are deactivated normally (specifying AP_DEACT_NORMAL) below this limit, new sessions are activated up to this limit. This must be set to a value in the range zero to 32767.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc	AP_OK
secondary_rc	AP_AS_SPECIFIED AP_AS_NEGOTIATED

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_EXCEEDS_MAX_ALLOWED
AP_INVALID_SET_NEGOTIABLE
AP_INVALID_PLU_NAME
AP_INVALID_MODE_NAME
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_MODE_NOT_RESET

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: Allocation Error

If the verb does not execute because of an allocation error, Personal Communications returns the following parameters:

primary_rc AP_ALLOCATION_ERROR

secondary_rc AP_ALLOCATION_FAILURE_NO_RETRY

sense_data Sense data associated with allocation error.

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

Returned Parameters: Other Errors

If the verb does not execute because of an error, Personal Communications returns the following parameters:

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

The RESET_SESSION_LIMIT verb requests that the mode session limits be reset.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_RESET_SESSION_LIMIT

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name LU name of the local LU requested to reset session limits. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias Alias of the local LU requested to reset session limits. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If this is set to all zeros, the verb is forwarded to the LU associated with the CP (the default LU).

plu_alias Alias by which the partner LU is known to the local LU. This name must match the name of a partner LU established during configuration. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. If this field is set to all zeros, the **fqplu_name** field is used to specify the required partner LU.

fqplu_name Fully qualified LU name for the partner LU. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is only significant if the **plu_alias** field is set to all zeros.

mode_name Name of a set of networking characteristics defined during configuration. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

mode_name_select Selects whether session limits should be reset on a single specified mode, or on all modes between the local and partner LUs.

AP_ONE
AP_ALL

set_negotiable Specifies whether the maximum negotiable session limit for this mode should be modified.

AP_YES
AP_NO

responsible Indicates whether the source (local) or target (partner) LU is responsible for deactivating sessions after the session limit is reset (AP_SOURCE or AP_TARGET).

drain_source Specifies whether the source LU satisfies waiting session requests before deactivating a session when session limits are changed or reset (AP_NO or AP_YES).

drain_target Specifies whether the target LU satisfies waiting session requests before deactivating a session when session limits are changed or reset (AP_NO or AP_YES).

force Specifies whether session limits will be set to zero even if CNOS negotiation fails (AP_YES or AP_NO).

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

secondary_rc
AP_FORCED
AP_AS_SPECIFIED
AP_AS_NEGOTIATED

Returned Parameters: Parameter Check

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

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

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_MODE_RESET

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: Allocation Error

If the verb does not execute because of an allocation error, Personal Communications returns the following parameters:

primary_rc AP_ALLOCATION_ERROR

secondary_rc AP_ALLOCATION_FAILURE_NO_RETRY

sense_data Sense data associated with allocation error.

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

Returned Parameters: Other Errors

If the verb does not execute because of an error, Personal Communications returns the following parameters:

primary_rc AP_CONV_FAILURE_NO_RETRY
AP_CNOS_PARTNER_LU_REJECT

secondary_rc AP_CNOS_COMMAND_RACE_REJECT
AP_CNOS_MODE_NAME_REJECT

Chapter 8. Node Operator Facility API Indications

The Node Operator Facility API generates indication verbs to notify a node operator about changes in the node. Indication verbs use the following general structure:

```
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;
```

© Copyright IBM Corp. 1989, 1996

DLC_INDICATION

This indication is generated when the DLC goes from active to inactive, or from inactive to active.

VCB Structure

```
typedef struct dlc_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     dlc_name[8];             /  link station name                   /
    unsigned char     description[RD_LEN]; /  resource description               /
   unsigned  char    reserva[2 ];             /   reserved                            /
} DLC_INDICATION;
```

Parameters

opcode AP_DLC_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc AP_OK

secondary_rc Equals zero.

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

deactivated Set to AP_YES when the DLC becomes inactive. Set to AP_NO when the DLC becomes active.

dlc_name Name of DLC. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

description Resource description (as specified on DEFINE_DLC). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

DLUS_INDICATION

This indication is generated when a pipe to a DLUS node goes from inactive to active (or vice-versa). Pipe statistics are supplied when the pipe becomes inactive.

VCB Structure

```
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[2 ];             / 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;
```

Parameters

opcode AP_DLUS_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc
AP_OK

secondary_rc
Equals zero.

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

deactivated
Set to AP_YES when the pipe becomes inactive. Set to AP_NO when the pipe becomes active.

dlus_name
Name of DLUS. This is a 17-byte string composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, which is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

pipe_stats.reqactpu_sent
Number of REQACTPUs sent to DLUS over the pipe.

pipe_stats.reqactpu_rsp_received
Number of RSP(REQACTPU)s received from DLUS over the pipe.

pipe_stats.actpu_received
Number of ACTPUs received from DLUS over the pipe.

pipe_stats.actpu_rsp_sent
Number of RSP(ACTPU)s sent to DLUS over the pipe.

pipe_stats.reqdactpu_sent
Number of REQDACTPUs sent to DLUS over the pipe.

pipe_stats.reqdactpu_rsp_received
Number of RSP(REQDACTPU)s received from DLUS over the pipe.

pipe_stats.dactpu_received
Number of DACTPUs received from DLUS over the pipe.

pipe_stats.dactpu_rsp_sent
Number of RSP(DACTPU)s sent to DLUS over the pipe.

pipe_stats.actlu_received
Number of ACTLUs received from DLUS over the pipe.

pipe_stats.actlu_rsp_sent
Number of RSP(ACTLU)s sent to DLUS over the pipe.

pipe_stats.dactlu_received
Number of DACTLUs received from DLUS over the pipe.

pipe_stats.dactlu_rsp_sent
Number of RSP(DACTLU)s sent to DLUS over the pipe.

pipe_stats.sscp_pu_mus_rcvd
Number of SSCP-PU MUs received from DLUS over the pipe.

pipe_stats.sscp_pu_mus_sent
Number of SSCP-PU MUs sent to DLUS over the pipe.

pipe_stats.sscp_lu_mus_rcvd
Number of SSCP-LU MUs received from DLUS over the pipe.

pipe_stats.sscp_lu_mus_sent
Number of SSCP-LU MUs sent to DLUS over the pipe.

FOCAL_POINT_INDICATION

This indication is generated whenever a focal point is acquired, changed or revoked.

VCB Structure

```
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[2 ];         /  reserved                        /
} FOCAL_POINT_INDICATION;
```

Parameters

opcode AP_FOCAL_POINT_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc AP_OK

secondary_rc Equals zero..

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

ms_category Category of focal point where the focal point has been acquired, changed or revoked. This may either be one of the four-byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services categories as described in SNA Management Services, or an 8-byte type 1134 EBCDIC installation defined name.

fp_fqcp_name The fully qualified control point name of the current focal point. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This name will be all zeros if the focal point has been revoked and not replaced (so that there is no currently active focal point).

FOCAL_POINT_INDICATION

ms_appl_name Name of the current focal point application. This may either be one of the four-byte architecturally defined values (right-padded with EBCDIC spaces) for Management Services applications as described in SNA Management Services, or an 8-byte type-1134 EBCDIC installation defined name. This will be all zeros if the focal point has been revoked and not replaced (so that there is no currently active focal point).

fp_type Type of focal point. Refer to SNA Management Services for further detail.

AP_EXPLICIT_PRIMARY_FP
AP_BACKUP_FP
AP_DEFAULT_PRIMARY_FP
AP_DOMAIN_FP
AP_HOST_FP
AP_NO_FP

fp_status Status of the focal point:

AP_NOT_ACTIVE
The focal point has gone from active to inactive.

AP_ACTIVE
The focal point has gone from inactive or pending active to active.

fp_routing Type of routing that applications should specify when using MDS transport to send data to the focal point (only significant if the focal point status is AP_ACTIVE):

AP_DEFAULT
Default routing is used to deliver the MDS_MU to the focal point.

AP_DIRECT
The MDS_MU will be routed on a session directly to the focal point.

LOCAL_TOPOLOGY_INDICATION

This indication is generated when a TG entry in a node's local topology database changes from active to inactive, or from inactive to active.

VCB Structure

```
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    reserva[2 ];     / reserved                     /
} LOCAL_TOPOLOGY_INDICATION;
```

Parameters

opcode AP_LOCAL_TOPOLOGY_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc AP_OK

secondary_rc Equals zero..

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

status Specifies the status of the TG. This may be one or more of the following ORed together:

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_NONE

dest Fully qualified destination node name for the TG. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

dest_type Type of the node. It is one of the following:

AP_END_NODE
AP_NETWORK_NODE
AP_VRN

tg_num Number associated with the TG.

LS_INDICATION

This indication is generated when the number of active sessions using the link changes, or the external state of the link station changes. Link station statistics are supplied when the link station becomes inactive.

VCB Structure

```
typedef struct ls_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    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    reserva[19];             /   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;
```

Parameters

opcode AP_LS_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc
AP_OK

secondary_rc
Equals zero.

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

deactivated
Set to AP_YES when the LS becomes inactive. Set to AP_NO when the LS becomes active.

ls_name Name of link station. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

description
Resource description (as specified on DEFINE_LS). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

adj_cp_name
Fully-qualified, 17-byte long, adjacent CP name. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

adj_node_type
Type of the node. It is one of the following:

AP_END_NODE
AP_NETWORK_NODE
AP_LEN_NODE
AP_VRN

act_sess_count
Total number of active sessions (both endpoint and intermediate) using the link.

indication_cause
Cause of the indication. It is one of the following:

AP_ACTIVATION_STARTED
The link is activating.

AP_ACTIVATING
The link has become active.

AP_DEACTIVATION_STARTED
The link is being deactivated.

AP_DEACTIVATING
The link has become inactive.

AP_SESS_COUNT_CHANGING
The number of active sessions using the link has changed.

AP_FAILED
The link has failed.

AP_ACTIVATION_FAILED
The link failed to activate.

AP_DATA_LOST
A previous indication has been lost. Note that link station statistics are only supplied when the link station goes from active to inactive (that is, deactivating is set to AP_YES and **indication_cause** is set to AP_DEACTIVATING). In all other cases the fields are reserved.

ls_stats.in_xid_bytes
Total number of XID (Exchange Identification) bytes received on this link station.

ls_stats.in_msg_bytes
Total number of data bytes received on this link station.

ls_stats.in_xid_frames
Total number of XID (Exchange Identification) frames received on this link station.

ls_stats.in_msg_frames
Total number of data frames received on this link station.

ls_stats.out_xid_bytes
Total number of XID (Exchange Identification) bytes sent on this link station.

ls_stats.out_msg_bytes
Total number of data bytes sent on this link station.

ls_stats.out_xid_frames
Total number of XID (Exchange Identification) frames sent on this link station.

ls_stats.out_msg_frames
Total number of data frames sent on this link station.

ls_stats.in_invalid_sna_frames
Total number of SNA invalid frames received on this link station.

ls_stats.in_session_control_frames
Total number of session control frames received on this link station.

ls_stats.out_session_control_frames
Total number of session control frames sent on this link station.

ls_stats.echo_rsps
Number of echo responses received from the adjacent node. Echo requests are sent periodically to gauge the propagation delay to the adjacent node.

ls_stats.current_delay
Time (in milliseconds) that it took for the last test signal to be sent and returned from this link station to the adjacent link station.

ls_stats.max_delay
Longest time taken (in milliseconds) for a test signal to be sent and returned from this link station to the adjacent link station.

ls_stats.min_delay
Shortest time taken (in milliseconds) for a test signal to be sent and returned from this link station to the adjacent link station.

ls_stats.max_delay_time
Time since system startup (in hundredths of a second) when the longest delay occurred.

ls_stats.good_xids
Total number of successful XID exchanges that have occurred on this link station since it was started.

ls_stats.bad_xids
Total number of unsuccessful XID exchanges that have occurred on this link station since it was started.

tg_num Number associated with the TG.

sense_data
This sense data is set if Personal Communications detects an XID protocol error. This field is reserved unless **indication_cause** is AP_FAILED.

LU_0_TO_3_INDICATION

This indication is generated when the state of a local LU (Type 0-3) changes.

VCB Structure

```
typedef struct lu_ _to_3_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    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     /
} LU_ _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    reserve;                /  reserved                      /
} SESSION_STATS;
```

Parameters

opcode AP_LU_0_TO_3_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc
AP_OK

secondary_rc
Equals zero.

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

pu_name Name of local PU. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

lu_name Name of the local LU whose state has changed. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

description
Resource description (as specified on DEFINE_LU_0_TO_3). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

nau_address
Network addressable unit address of the LU (which will be in the range 10—2554).

lu_sscp_sess_active
Specifies whether the ACTLU has been successfully processed (AP_YES or AP_NO).

appl_conn_active
Set if the application is using this LU (AP_YES or AP_NO).

plu_sess_active
Specifies whether the PLU-SLU session has been activated (AP_YES or AP_NO).

host_attachment
Specifies the LU host attachment type:

AP_DLUR_ATTACHED
LU is attached to host system using DLUR.

AP_DIRECT_ATTACHED
LU is directly attached to host system. Note the LU-SSCP and PLU-SLU statistics are only valid when the sessions go from active to inactive. In all other cases the fields are reserved.

lu_sscp_stats.rcv_ru_size
This field is always reserved.

lu_sscp_stats.send_ru_size
This field is always reserved.

lu_sscp_stats.max_send_btu_size
Maximum BTU size that can be sent.

lu_sscp_stats.max_rcv_btu_size
Maximum BTU size that can be received.

lu_sscp_stats.max_send_pac_win
This field will always be set to zero.

lu_sscp_stats.cur_send_pac_win
This field will always be set to zero.

lu_sscp_stats.max_rcv_pac_win
This field will always be set to zero.

lu_sscp_stats.cur_rcv_pac_win
This field will always be set to zero.

lu_sscp_stats.send_data_frames
Number of normal flow data frames sent.

lu_sscp_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

lu_sscp_stats.send_data_bytes
Number of normal flow data bytes sent.

lu_sscp_stats.rcv_data_frames
Number of normal flow data frames received.

lu_sscp_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

lu_sscp_stats.rcv_data_bytes
Number of normal flow data bytes received.

lu_sscp_stats.sidh
Session ID high byte.

lu_sscp_stats.sidl
Session ID low byte.

lu_sscp_stats.odai
Origin Destination Address Indicator. When bringing up a session, the sender of the ACTLU sets this field to zero if the local node contains the primary link station, and sets it to one if the ACTLU sender is the node containing the secondary link station.

lu_sscp_stats.ls_name
Link station name associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant. This field can be used to correlate this session with the link over which the session flows.

plu_stats.rcv_ru_size
Maximum receive RU size.

plu_stats.send_ru_size
Maximum send RU size.

plu_stats.max_send_btu_size
Maximum BTU size that can be sent.

plu_stats.max_rcv_btu_size
Maximum BTU size that can be received.

plu_stats.max_send_pac_win
Maximum size of the send pacing window on this session.

plu_stats.cur_send_pac_win
Current size of the send pacing window on this session.

plu_stats.max_rcv_pac_win
Maximum size of the receive pacing window on this session.

plu_stats.cur_rcv_pac_win
Current size of the receive pacing window on this session.

plu_stats.send_data_frames
Number of normal flow data frames sent.

plu_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

plu_stats.send_data_bytes
Number of normal flow data bytes sent.

plu_stats.rcv_data_frames
Number of normal flow data frames received.

plu_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

plu_stats.rcv_data_bytes
Number of normal flow data bytes received.

plu_stats.sidh
Session ID high byte.

plu_stats.sidl
Session ID low byte.

plu_stats.odai
Origin Destination Address Indicator. When bringing up a session, the sender of the ACTLU sets this field to zero if the local node contains the primary link station, and sets it to one if the ACTLU sender is the node containing the secondary link station.

plu_stats.ls_name
Link station name associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant. This field can be used to correlate this session with the link over which the session flows.

MODE_INDICATION

This indication is sent when a local LU and partner LU combination start to use a particular mode, and when the current session count for the local LU, partner LU, and mode combination changes.

VCB Structure

```
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[2 ];            / reserved                      /
} MODE_INDICATION;
```

Parameters

opcode AP_MODE_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc AP_OK

secondary_rc Equals zero..

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

removed Specifies whether an entry is being removed (AP_YES or AP_NO). It is set when entry is being removed rather than added.

lu_alias Locally defined LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

plu_alias Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

fqplu_name 17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

mode_name Mode name which designates the network properties for a group of sessions. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

description Resource description (as specified on DEFINE_MODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

curr_sess_count Current count of sessions for this local LU,partner LU, and mode combination.

PLU_INDICATION

This indication is generated when a local LU first connects to a partner LU. This will happen when the first ALLOCATE to this PLU is processed or when the first BIND is received from this PLU. This indication is also generated if the partner CP name changes.

VCB Structure

```
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[2 ];             / reserved                     /
} PLU_INDICATION;
```

Parameters

opcode AP_PLU_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc AP_OK

secondary_rc Equals zero..

data_lost Specifies whether one or more indications have been lost (AP_YES or AP_NO). It is set when an internal component was unable to send a previous indication. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

removed Specifies whether an entry is being removed (AP_YES or AP_NO). It is set when entry is being removed rather than added.

lu_alias Locally defined LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

plu_alias Partner LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

fqplu_name 17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with

EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

description Resource description (as specified on DEFINE_PARTNER_LU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

partner_cp_name 17-byte fully qualified network name for the control point of the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

partner_lu_located Specifies whether the partner CP name has been resolved (AP_YES or AP_NO), and thus whether the **partner_cp_name** field contains the CP name.

PORT_INDICATION

This indication is generated when the port goes from active to inactive (or vice-versa).

VCB Structure

```
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 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[2 ];             /   reserved                            /
} PORT_INDICATION;
```

Parameters

opcode AP_PORT_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc AP_OK

secondary_rc Equals zero.

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

deactivated Set to AP_YES when the port becomes inactive. Set to AP_NO when the port becomes active.

port_name Name of port. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

description Resource description (as specified on DEFINE_PORT). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

PU_INDICATION

This indication is generated when the state of a local PU changes.

VCB Structure

```
typedef struct 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    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      /
} 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 win size     /
    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   reserve;                 / reserved                          /
} SESSION_STATS;
```

Parameters

opcode AP_PU_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc
AP_OK

secondary_rc
Equals zero.

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

pu_name Name of the PU (configured on the DEFINE_LS verb). This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

description
Resource description (as specified on DEFINE_LS or DEFINE_INTERNAL_PU). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

pu_sscp_sess_active
Specifies whether the ACTPU has been successfully processed (AP_YES or AP_NO).

host_attachment
PU host attachment type:

AP_DLUR_ATTACHED
PU is attached to host system using DLUR.

AP_DIRECT_ATTACHED
PU is directly attached to host system.

Note: PU-SSCP statistics are valid only when the session state has moved from active to inactive. In all other cases the following fields are reserved:

pu_sscp_stats.rcv_ru_size
This field is always reserved.

pu_sscp_stats.send_ru_size
This field is always reserved.

pu_sscp_stats.max_send_btu_size
Maximum BTU size that can be sent.

pu_sscp_stats.max_rcv_btu_size
Maximum BTU size that can be received.

pu_sscp_stats.max_send_pac_win
This field will always be set to zero.

pu_sscp_stats.cur_send_pac_win
This field will always be set to zero.

pu_sscp_stats.max_rcv_pac_win
This field will always be set to zero.

pu_sscp_stats.cur_rcv_pac_win
This field will always be set to zero.

pu_sscp_stats.send_data_frames
Number of normal flow data frames sent.

pu_sscp_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

pu_sscp_stats.send_data_bytes
Number of normal flow data bytes sent.

pu_sscp_stats.rcv_data_frames
Number of normal flow data frames received.

pu_sscp_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

pu_sscp_stats.rcv_data_bytes
Number of normal flow data bytes received.

pu_sscp_stats.sidh
Session ID high byte.

pu_sscp_stats.sidl
Session ID low byte.

pu_sscp_stats.odai
Origin Destination Address Indicator. When bringing up a session, the sender of the ACTPU sets this field to zero if the local node contains the primary link station, and sets it to one if the ACTPU sender is the node containing the secondary link station.

pu_sscp_stats.ls_name
Link station name associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant. This field can be used to correlate this session with the link over which the session flows.

REGISTRATION_FAILURE

REGISTRATION_FAILURE indicates that an attempt to register resources with the Network Node server failed.

VCB Structure

```
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[2 ];            / reserved                      /
} REGISTRATION_FAILURE;
```

Parameters

opcode AP_REGISTRATION_FAILURE

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc AP_OK

resource_name Name of resource that failed to register. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

resource_type Resource type. One of the following:

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

description Resource description (as specified on DEFINE_LOCAL_LU, or DEFINE_ADJACENT_LEN_NODE). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

sense_data Sense data (specified in SNA Formats).

RTP_INDICATION

This indication is generated when an RTP Connection is connected or disconnected the active session count changes the connection performs a path-switch.

When the connection is disconnected, final RTP statistics will be returned. At other times the **rtp_stats** field is reserved.

VCB Structure

```
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[4 ];             / reserved                           /
} RTP_STATISTICS;
```

Parameters

opcode AP_RTP_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc
AP_OK

secondary_rc
Equals zero.

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then the data contained may have changed more than once since the previous indication received.

connection_state
The current state of the RTP connection. It is one of the following:

AP_CONNECTING
Connection setup has started, but is not yet complete.

AP_CONNECTED
The connection is fully active.

AP_DISCONNECTED
The connection is no longer active.

rtp_name
RTP connection name. This name is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

num_sess_active
Number of sessions currently active on the connection.

indication_cause
Cause of the indication. It is one of the following:

AP_ACTIVATED
The connection has become active.

AP_DEACTIVATED
The connection has become inactive.

AP_PATH_SWITCHED
The connection has successfully completed a path switch.

AP_SESS_COUNT_CHANGING
The number of active sessions using the connection has changed.

AP_SETUP_FAILED
The connection has failed before becoming fully active. Note that RTP connection statistics are only supplied when the connection becomes inactive, that is when **indication_cause** is AP_DEACTIVATED or AP_SETUP_FAILED. In all other cases the fields are reserved.

rtp_stats.bytes_sent
Total number of bytes that the local node has sent on this RTP connection.

rtp_stats.bytes_received
Total number of bytes that the local node has received on this RTP connection.

rtp_stats.bytes_resent
Total number of bytes resent by the local node owing to loss in transit.

rtp_stats.bytes_discarded
Total number of bytes sent by the other end of the RTP connection which were discarded as duplicates of data already received.

rtp_stats.packets_sent
Total number of packets that the local node has sent on this RTP connection.

rtp_stats.packets_received
Total number of packets that the local node has received on this RTP connection.

rtp_stats.packets_resent
Total number of packets resent by the local node owing to loss in transit.

rtp_stats.packets_discarded
Total number of packets sent by the other end of the RTP connection which were discarded as duplicates of data already received.

rtp_stats.gaps_detected
Total number of gaps detected by the local node. Each gap corresponds to one or more lost frames.

rtp_stats.send_rate
Current send rate on this RTP connection (measured in kilobits per second). This is the maximum allowed send rate as calculated by the ARB algorithm.

rtp_stats.max_send_rate
Maximum send rate on this RTP connection (measured in kilobits per second).

rtp_stats.min_send_rate
Minimum send rate on this RTP connection (measured in kilobits per second).

rtp_stats.receive_rate
Current receive rate on this RTP connection (measured in kilobits per second). This is the actual receive rate calculated over the last measurement interval.

rtp_stats.max_receive_rate
Maximum receive rate on this RTP connection (measured in kilobits per second).

rtp_stats.min_receive_rate
Minimum receive rate on this RTP connection (measured in kilobits per second).

rtp_stats.burst_size
Current burst-size on the RTP Connection measured in bytes.

rtp_stats.up_time
Total number of seconds the RTP connection has been active.

rtp_stats.smooth_rtt
Smoothed measure of round-trip time between the local node and the partner RTP node (measured in milliseconds).

rtp_stats.last_rtt
The last measured round-trip time between the local node and the partner RTP node (measured in milliseconds).

rtp_stats.short_req_timer
The current duration used for the SHORT_REQ timer (measured in milliseconds).

rtp_stats.short_req_timeouts
Total number of times the SHORT_REQ timer has expired for this RTP connection.

rtp_stats.liveness_timeouts
Total number of times the liveness timer has expired for this RTP connection. The liveness timer expires when the connection has been idle for the period specified in **rtp_connection_detail.liveness_timer**.

rtp_stats.in_invalid_sna_frames
Total number of SNA frames received and discarded as invalid on this RTP connection.

rtp_stats.in_sc_frames
Total number of session control frames received on this RTP connection.

rtp_stats.out_sc_frames
Total number of session control frames sent on this RTP connection.

SESSION_INDICATION

This indication is generated when a session is activated or deactivated. When a session is deactivated, final session statistics will be returned. When a session is activated, the **sess_stats** field is reserved.

VCB Structure

```
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   reserv1;                 / reserved                            /
    SESSION_STATS    sess_stats;              / session  statistics                 /
                                                / correlator ID                       /
    unsigned  char   reserva[2 ];             / 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 win size      /
    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    reserve;                /   reserved                       /
} SESSION_STATS;
```

Parameters

opcode AP_SESSION_INDICATION

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

primary_rc
AP_OK

secondary_rc
Equals zero.

data_lost Specifies whether data has been lost (AP_YES or AP_NO). It is set when an internal component detects a failure which has caused a previous indication to be lost. If the **data_lost** flag is set to AP_YES then subsequent data fields may be set to null. The application should issue a QUERY verb to update the information which has been lost.

deactivated
Set to AP_NO when a session is activated. Set to AP_YES when a session is deactivated.

lu_name LU name. This name is an 8-byte type-A EBCDIC character string.

lu_alias Locally defined LU alias. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant.

plu_alias Partner LU alias. This is an 8-byte string in a locally displayable character set.

fqplu_name
17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

mode_name
Mode name which designates the network properties for a group of sessions. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

session_id
8-byte identifier of the session.

fqpcid.pcid
Procedure Correlator ID. This is an 8-byte hexadecimal string.

fqpcid.fqcp_name
Fully qualified CP name. This name is 17 bytes long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

sense_data
The sense data sent or received on the UNBIND request. This field is reserved if **deactivated** is AP_NO.

sess_stats.rcv_ru_size
Maximum receive RU size.

sess_stats.send_ru_size
Maximum send RU size.

sess_stats.max_send_btu_size
Maximum BTU size that can be sent.

sess_stats.max_rcv_btu_size
Maximum BTU size that can be received.

sess_stats.max_send_pac_win
Maximum size of the send pacing window on this session.

sess_stats.cur_send_pac_win
Current size of the send pacing window on this session.

sess_stats.max_rcv_pac_win
Maximum size of the receive pacing window on this session.

sess_stats.cur_rcv_pac_win
Current size of the receive pacing window on this session.

sess_stats.send_data_frames
Number of normal flow data frames sent.

sess_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

sess_stats.send_data_bytes
Number of normal flow data bytes sent.

sess_stats.rcv_data_frames
Number of normal flow data frames received.

sess_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

sess_stats.rcv_data_bytes
Number of normal flow data bytes received.

sess_stats.sidh
Session ID high byte.

sess_stats.sidl
Session ID low byte.

sess_stats.odai
Origin Destination Address Indicator. When bringing up a session, the sender of the BIND sets this field to zero if the local node contains the primary link station, and sets it to one if the BIND sender is the node containing the secondary link station.

sess_stats_stats.ls_name
Link station name associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant. This field can be used to correlate the session statistics with the link over which session traffic flows.

SESSION_INDICATION

Chapter 9. Security Verbs

This chapter describes verbs used to define and delete security passwords.

© Copyright IBM Corp. 1989, 1996

DEFINE_LU_LU_PASSWORD

DEFINE_LU_LU_PASSWORD provides a password which is used for session level verification between a local LU and a partner LU.

VCB Structure

```
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    reserv4;             / reserved                     /
    unsigned char    description[RD_LEN];
                                          / resource description         /
    unsigned char    reserv3[8];          / reserved                     /
    unsigned char    password[8];         / password                     /
} DEFINE_LU_LU_PASSWORD;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_LU_LU_PASSWORD

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name LU name of the local LU. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias Local LU alias. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_alias** and the **lu_name** are set to all zeros, the verb is forwarded to the LU associated with the CP (the default LU).

fqplu_name Fully qualified partner LU name. This name is 17-byte s long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

description Resource description.

password Password. This is an 8-byte hexadecimal string. Note that the least significant bit of each byte in the password is not used in session-level verification.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_PLU_NAME
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_USERID_PASSWORD

DEFINE_USERID_PASSWORD defines a password associated with a User ID.

VCB Structure

```
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[1 ];            / user id                      /
   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[1 ];            /  password                    /
   unsigned  char    profiles[1 ][1 ];        /  profiles                    /
} USERID_PASSWORD_CHARS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_USERID_PASSWORD

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

define_type Specifies the type of user password being defined:

AP_ADD_USER
Specifies a new user, or change of password for an existing user.

AP_ADD_PROFILES
Specifies an addition to the profiles for an existing user.

user_id User identifier. This is a 10-byte type-AE EBCDIC character string, padded to the right with EBCDIC spaces.

password_chars.description Resource description. This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

password_chars.profile_count Number of profiles.

password_chars.password User's password. This is a 10-byte type-AE EBCDIC character string, padded to the right with EBCDIC spaces.

password_chars.profiles Profiles associated with user. Each of these is a 10-byte type-AE EBCDIC character string, padded to the right with EBCDIC spaces.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_NO_PROFILES
AP_UNKNOWN_USER
AP_INVALID_UPDATE_TYPE
AP_TOO_MANY_PROFILES
AP_INVALID_USERID
AP_INVALID_PROFILE
AP_INVALID_PASSWORD

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LU_LU_PASSWORD

DELETE_LU_LU_PASSWORD deletes an LU-LU password.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_LU_LU_PASSWORD

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

lu_name LU name of the local LU. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias Local LU alias. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_alias** and the **lu_name** are set to all zeros, the verb is forwarded to the LU associated with the CP (the default LU).

fqplu_name Fully qualified partner LU name. This name is 17-byte s long and is right-padded with EBCDIC spaces. It is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc	AP_PARAMETER_CHECK
secondary_rc	AP_INVALID_PLU_NAME AP_INVALID_LU_NAME AP_INVALID_LU_ALIAS

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc	AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc	AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

DELETE_USERID_PASSWORD

DELETE_USERID_PASSWORD deletes a password associated with a User ID.

VCB Structure

```
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[1 ];            / 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[1 ];           /  password                     /
   unsigned  char    profiles[1 ][1 ];       /  profiles                     /
} USERID_PASSWORD_CHARS;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_USERID_PASSWORD

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

delete_type
Specifies the type of delete:

AP_REMOVE_USER
Deletes the user, password and all associated profiles.

AP_REMOVE_PROFILES
Deletes the specified profiles.

user_id User identifier. This is a 10-byte type-AE EBCDIC character string, padded to the right with EBCDIC spaces.

password_chars.description
Reserved.

password_chars.profile_count
Number of profiles.

password_chars.password
This field is ignored when processing this verb.

password_chars.profiles
Profiles associated with user. Each of these is a 10-byte type-AE EBCDIC character string, padded to the right with EBCDIC spaces.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_NO_PROFILES
AP_UNKNOWN_USER
AP_INVALID_UPDATE_TYPE

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_USERID_PASSWORD

Chapter 10. APING and CPI-C Verbs

This chapter describes verbs used to “ping” another node and verbs used to define, delete, and query CPI-C side information.

© Copyright IBM Corp. 1989, 1996

APING

APING allows a management application to “ping” a remote LU in the network. A verification data string (of specified length) can be appended to the end of the VCB and returned when the **partner_ver_len** field is set to a value greater than zero.

Personal Communications APING is implemented as an internal “service TP” which uses the Personal Communications APPC API (described in the *Personal Communications Client Server Communications*). Note, however, that APING's internal use of this API does not require the interface to be exposed to external applications.

VCB Structure

```
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[1 ];             / password                         /
    unsigned char    user_id[1 ];         / 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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_APING

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

lu_name LU name of the local LU from which the APING verb is sent. This name is an 8-byte type-A EBCDIC character string. If this field is set to all zeros, the **lu_alias** field will be used for determining the local LU.

lu_alias Alias for the local LU from which the APING verb is sent. This is an 8-byte string in a locally displayable character set. This field is only significant if the **lu_name** field is set to all zeros, in which case all 8 bytes are significant and must be set. If both the **lu_name** and the **lu_alias** are set to binary zeros then the default (Control Point) LU is used.

plu_alias Alias by which the partner LU is known to the local TP. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant and must be set. This name must match the name of a partner LU established during configuration. If this parameter is set to binary zeros, the **fqplu_name** parameter is used instead.

mode_name Name of the mode to be used. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

tp_name Name of the invoked TP. This is a 64-byte string, padded to the right with spaces. The string is usually set to "APINGD."

security Specifies the information the partner LU requires in order to validate access to the invoked TP:

AP_NONE
AP_PGM
AP_SAME

pwd Password associated with **user_id**. This is a 10-byte type-AE EBCDIC character string, padded to the right with EBCDIC spaces. Only needed if **security** is set to AP_PGM.

user_id User ID required to access the partner TP. This is a 10-byte type-AE EBCDIC character string, padded to the right with EBCDIC spaces. Needed if **security** is set to AP_PGM or AP_SAME.

dlen Length of data to be sent by APING TP. APING sends a string of zeros, of length **dlen**.

consec Number of consecutive sends performed during each iteration. APING issues this number of MC_SEND_DATA verbs, each consisting of **dlen** bytes of data. If the **echo** parameter is set to AP_YES, APING marks the last MC_SEND_DATA as AP_SEND_DATA_P_TO_R_FLUSH (Prepare to Receive Flush) and awaits a response containing data from the partner APINGD TP (by issuing a MC_RECEIVE_AND_WAIT). If the **echo** parameter is set to AP_NO, APING flushes the data and awaits a confirm (by marking the last MC_SEND_DATA as AP_SEND_DATA_CONFIRM). In either case, the sequence described here corresponds to an SNA chain.

fqplu_name 17-byte fully qualified network name for the partner LU. This name is composed of two type-A EBCDIC character strings concatenated by an EBCDIC dot, and is right-padded with EBCDIC spaces. (Each name can have a maximum length of 8 bytes with no embedded spaces.) This field is only significant if the **plu_alias** field is set to all zeros.

echo Specifies whether the APING TP expects a response when it has completed sending the required amount of data:

AP_YES
AP_NO

iterations Number of iterations of consecutive sequences (defined by the **consec** parameter) issued by APING. In SNA terms, this parameter defines the number of chains which will be sent.

partner_ver_len Maximum length of the partner TP verification data string which can be received by the management application.

Returned Parameters: Successful execution

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

primary_rc AP_OK

sense_data This will be zero if the verb has returned successfully.

alloc_time Time required (in milliseconds) for the MC_ALLOCATE to the remote TP to complete.

min_time Minimum time (in milliseconds) required for a data-sending iteration. This parameter includes the time required for the partner to respond (either by sending data or issuing a confirm, depending on the setting of the **echo** parameter).

avg_time Average time (in milliseconds) required for a data-sending iteration. This parameter includes the time required for the partner to respond (either by sending data or issuing a confirm, depending on the setting of the **echo** parameter).

max_time Maximum time (in milliseconds) required for a data-sending iteration. This parameter includes the time required for the partner to respond (either by sending data or issuing a confirm, depending on the setting of the **echo** parameter).

partner_ver_len Length of verification string returned by the partner TP. The string itself is appended to the end of the VCB.

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

Returned Parameters: Unsuccessful Execution

APING uses the MC_ALLOCATE, MC_SEND_DATA, MC_RECEIVE_AND_WAIT, MC_CONFIRM and MC_DEALLOCATE verbs provided by the Personal Communications APPC API. The parameters returned by these verbs in the case of unsuccessful execution are documented in the *Personal Communications Client Server Communications*.

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

CPI-C Verbs

DEFINE_CPIC_SIDE_INFO

This verb adds or replaces a side information entry in memory. A CPI-C side information entry associates a set of conversation characteristics with a symbolic destination name. If there is already a side information entry in memory with the same symbolic destination name as the one supplied with this verb, it is overwritten with the data supplied to this call. See *Personal Communications CPI-C Specification* for more information about the CPI-C support provided by Personal Communications.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DEFINE_CPIC_SIDE_INFO

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

sym_dest_name
Symbolic destination name which identifies the side information entry. This is up to 8 bytes long, padded with spaces, in the locally displayable character set. The allowed characters are the uppercase letters (A to Z) and the digits 0-9.

def_data.description
Resource description (returned on QUERY_CPIC_SIDE_INFO). This is a 16-byte string in a locally displayable character set. All 16 bytes are significant.

def_data.side_info.partner_lu_name
Fully qualified name of the partner LU. This name is 17 bytes long and is right-padded with spaces, in the locally displayable character set. It is composed of two character strings concatenated by a dot. (Each name can have a maximum length of 8 bytes with no embedded spaces.)

def_data.side_info.tp_name_type
TP name type. This field is set to one of the following:

XC_APPLICATION_TP
Specifies that the TP name supplied is not a service TP. All characters specified in the TP name must be valid characters in the locally displayable character set.

XC_SNA_SERVICE_TP
Specifies that the TP name supplied is that of a service TP. All characters, except the first, specified in the TP must be valid characters in the locally displayable character set. The first character must be a hexadecimal digit in the range X'01' to X'3F', excluding X'0E' and X'0F'.

def_data.side_info.tp_name
TP name, a 64-byte character string in the locally displayable character set, right-padded with spaces.

def_data.side_info.mode_name
Mode name, an 8-byte character string in the locally displayable character set, padded to the right with spaces.

def_data.side_info.conversation_security_type
Conversation security type. This field is set to one of the following:

XC_SECURITY_NONE
XC_SECURITY_SAME
XC_SECURITY_PROGRAM.

def_data.side_info.security_user_id
User ID. Personal Communications will use this field for enforcing conversation-level security.

def_data.side_info.security_password
Password. Personal Communications will use this field for enforcing conversation-level security.

def_data.user_data
User data. This data is returned on QUERY_CPIC_SIDE_INFO but not used or interpreted by Personal Communications.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

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

primary_rc AP_PARAMETER_CHECK

secondary_rc AP_INVALID_SYM_DEST_NAME
AP_INVALID_LENGTH

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

DELETE_CPIC_SIDE_INFO

This verb deletes a CPI-C side information entry. See the Personal Communications CPI-C Specification for more information about the CPI-C support provided by Personal Communications.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DELETE_CPIC_SIDE_INFO

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

sym_dest_name Symbolic destination name which identifies the side information entry. This is up to 8 bytes long, padded with spaces, in the locally displayable character set. The allowed characters are the uppercase letters (A to Z) and the digits 0-9.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc AP_OK

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_INVALID_SYM_DEST_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_CPIC_SIDE_INFO

This verb returns the side information entry for a given symbolic destination name. The information is returned as a list. To obtain a specific side information entry, or a specific chunk of entries, the **sym_dest_name** field should be set. Otherwise this field should be set to all zeros.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_QUERY_CPIC_SIDE_INFO

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

buf_ptr Pointer to buffer into which list information can be written.

buf_size Size of buffer supplied. The data returned will not exceed this size.

num_entries Maximum number of entries to return. The number of entries will not exceed this value. A value of zero means no limit.

list_options This indicates what should be returned in the list information. The **sym_dest_name** specified (see below) represents an index value which is used to specify the starting point of the actual information to be returned:

AP_FIRST_IN_LIST
The index value is ignored and the returned list starts from the first entry in the list.

AP_LIST_FROM_NEXT
The returned list starts from the next entry in the list after the one specified by the supplied index value.

AP_LIST_INCLUSIVE
The returned list starts from the entry specified by the index value.

sym_dest_name Symbolic destination name which identifies the side information entry. This is up to 8 bytes long, padded with spaces, in the locally displayable character set. The allowed characters are the uppercase letters (A to Z) and the digits 0-9.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications returns the following parameters:

primary_rc
AP_OK

buf_size Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than **buf_size**.

num_entries
Number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than **num_entries**.

cpic_side_info_data.overlay_size
The number of bytes in this entry, and hence the offset to the next entry returned (if any).

cpic_side_info_data.sym_dest_name
Symbolic destination name for the returned side information entry.

cpic_side_info_data.def_data
Defined CPI-C side information as supplied on DEFINE_CPIC_SIDE_INFO verb. (Note CPIC calls may change the side information returned on this verb after the DEFINE_CPIC_SIDE_INFO has been processed by Personal Communications).

Returned Parameters: State Check

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

primary_rc AP_STATE_CHECK

secondary_rc AP_INVALID_SYM_DEST_NAME

Returned Parameters: Node Not Started

If the verb does not execute because the node has not yet been started, Personal Communications returns the following parameters:

primary_rc AP_NODE_NOT_STARTED

Returned Parameters: Node Stopping

If the verb does not execute because a STOP_NODE verb has been issued, Personal Communications returns the following parameters:

primary_rc AP_NODE_STOPPING

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

Chapter 11. Attach Manager Verbs

The Personal Communications Attach Manager is used to manage the launching of APPC or CPI-C programs. A description of the Attach Manager function is provided in **Personal Communications Client/Server Communications Programming**.

Personal Communications Node Operator Facility supports three verbs to control the Attach Manager. These verbs are available to any application program that uses Personal Communications Node Operator Facility.

© Copyright IBM Corp. 1989, 1996

DISABLE_ATTACH_MANAGER

The Personal Communications Attach Manager is enabled by default when the node is started. The user can issue this verb to disable all dynamic loading, This verb resets a global flag that the Attach Manager checks before launching a transaction program.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_DISABLE_ATTACH_MGR

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

Returned Parameters: Successful Execution

If the verb executes successfully, the Attach Manager returns the following parameters:

primary_rc AP_OK

Returned Parameters: Node not Started

If the verb does not execute because the Node has not yet been, the Attach Manager returns the following parameters.

primary_rc AP_NODE_NOT STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, the Attach Manager returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

ENABLE_ATTACH_MANAGER

If the Attach Manager has been disabled, it may be re-enabled by issuing the Personal Communications Node Operator Facility verb, ENABLE_AM. This sets a global flag that the Attach Manager checks before launching a Transaction Program.

VCB Structure

```
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
```

Supplied Parameters

The application supplies the following parameters:

opcode AP_ENABLE_ATTACH_MGR

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

Returned Parameters: Successful Execution

If the verb executes successfully, the Attach Manager returns the following parameters:

primary_rc AP_OK

Returned Parameters: Node not Started

If the verb does not execute because the Node has not yet been, the Attach Manager returns the following parameters.

primary_rc AP_NODE_NOT STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, the Attach Manager returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

QUERY_ATTACH_MANAGER

The QUERY_ATTACH_MANAGER verb can be used to discover the status of the Attach Manager component which can be started and stopped using the ENABLE_ATTACH_MANAGER and DISABLE_ATTACH_MANAGER commands.

VCB Structure

```
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;
```

Supplied Parameters

opcode AP_QUERY_ATTACH_MGR

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

Returned Parameters: Successful Execution

If the verb executes successfully, the following parameters are returned.

primary_rc AP_OK

active This field reports the status of the Attach Manager component:

AP_YES
The Attach Manager is active.

AP_NO
The Attach Manager is not active.

Returned Parameters: Parameter Check

If the verb does not execute because of a parameter error, the following parameters are returned.

primary_rc AP_PARAMETER_CHECK

Returned Parameters: Node not Started

If the verb does not execute because the Node has not yet been, the Attach Manager returns the following parameters.

primary_rc AP_NODE_NOT STARTED

Returned Parameters: System Error

If the verb does not execute because of a system error, the Attach Manager returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

Part 2. Personal Communications Management Services API

Chapter 12. Introduction to Management Services API 375
Management Services Verbs 375
Entry Points 375
Verb Control Blocks (VCB) 375
Compiling and Linking Your Management Services Program 376
DLL Interface 376
Using Personal Communications High-Level Language Header Files . . . 377
Common VCB Fields 377

Chapter 13. Management Services Entry Points 379
WinMS() 380
WinMSCleanup() 381
WinMSStartup() 382
WinMSRegisterApplication() 383
WinMSUnregisterApplication() 385
WinMSGetIndication() 387

Chapter 14. Management Services Verbs 389
TRANSFER_MS_DATA 390
MDS_MU_RECEIVED 393
SEND_MDS_MU 395
ALERT_INDICATION 398
FP_INDICATION 399
NMVT_RECEIVED 400

© Copyright IBM Corp. 1989, 1996

Chapter 12. Introduction to Management Services API

This part describes the Management Services API provided by Personal Communications.

Management Services Verbs

Personal Communications supports the following management services (MS) verbs, providing an application program with a method for reporting potential problems to MS focal points available in an SNA network.

ALERT_INDICATION
FP_INDICATION
MDS_MU_RECEIVED
NMVT_RECEIVED
SEND_MDS_MU
TRANSFER_MS_DATA

Entry Points

Personal Communications provides a library file that handles Management Services verbs.

Management Services verbs have a straightforward language interface. Your program fills in fields in a block of memory called a *verb control block* Then your program calls the entrypoint and passes a pointer to the verb control block. When its operation is complete, Management Services (MS) API returns, having used and then modified the fields in the verb control block. Your program can then read the returned parameters from the verb control block. The following is a list of entry points for Management Services verbs:

WinMS()
WinAsyncMS()
WinAsyncMSEx()
WinMSCancelAsyncRequest()
WinMSCleanup()
WinMSStartup()

See Chapter 3, “Node Operator Facility Entry Points” for detailed descriptions of the entry points.

Verb Control Blocks (VCB)

Programming Note: The base operating system optimizes performance by executing some subsystems in the calling application's address space. This means that incorrect use of local descriptor table (LDT) selectors by application programs that have not been fully or correctly debugged can cause improper operation, or perhaps system failures. Accordingly, application programs should not perform pointer arithmetic operations that involve changing the LDT selector field of a pointer.

© Copyright IBM Corp. 1989, 1996

The segment used for the verb control block (VCB) must be a read/write data segment. Your program can either declare the VCB as a variable in your program, allocate it or suballocate it from a larger segment. It must be sufficiently large to contain all the fields for the verb your program is issuing.

An application program should not change any part of the verb control block after it has been issued until the verb completes. When Management Services finishes the execution of a verb, it copies a complete, modified VCB back onto the original block. Therefore, if your program declares a verb control block as a variable, consider declaring it in static storage rather than on the stack of an internal procedure.

Fill all reserved and unused fields in each VCB with zeros (X'00'). In fact, it might be more time-efficient to set the entire verb control block to zeros before your program assigns the values to the parameters. Setting reserved fields to zeros is particularly important.

Note: If the VCB is not read/write, or if it is not at least 10 bytes (that is, large enough to hold the Management Services primary and secondary return codes), Management Services cannot access it, and the base operating system abnormally ends the process. This termination is recognized as a *general protection fault*, processor exception trap D.

Management Services returns the INVALID_VERB_SEGMENT primary return code when the VCB is too short or the incorrect type of segment is used.

Compiling and Linking Your Management Services Program

The WINMS32.LIB file can be found in the LIB subdirectory of the Personal Communications directory.

Save the source files for all the elements of your program. Later releases of Personal Communications might require that the program be recompiled and relinked to use the new functions in those releases.

You can create a program module (an .EXE file) to run on the base operating system by linking your compiled source files with the LINK utility program supplied with the base operating system. For more information on how to create a .EXE file, refer to the programming language reference manual associated with the language you are using.

The LINK program uses the compiled source files, a list of library files, and module definition files that contain information about your program. From these files, the LINK program builds the .EXE file that runs your program.

DLL Interface

Windows 95 provides two ways to dynamically link to a DLL:

Load-time linking
Run-time linking

To link to WINMS32.DLL at load time, you should link with C:\Personal_Communications\lib\WINMS32.LIB.

The WINMS32.LIB library contains entry point linkage information for all Personal Communications API entry points.

Run-time linking provides conditional linking to a DLL. In order to find the function address of the MS API entry point, your application can issue **LoadLibrary** for the module name of "WINMS32" and then **GetProcAddress** for the address of the routines for the 32-bit MS interface.

An application can use **FreeLibrary** to unlink after the application has finished using the MS interface.

Using Personal Communications High-Level Language Header Files

The C language is supported. Only 32 bit API support is provided.

To simplify the coding of Management Services verbs, the VCB structure for each verb, along with appropriate parameter constants, is supplied with Personal Communications. The pseudonym file WINMS.H is provided to assist in the development of programs. It contains the constant declarations and data types for each supplied and returned parameter in the NOF calls, and call prototypes. This file can be found in the INCLUDE subdirectory of the Personal Communications directory.

Common VCB Fields

This chapter documents the syntax of each verb passed across the Management Services API. It also describes the parameters passed in and returned for each verb.

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

Each VCB has a number of common fields. These are listed and described below.

opcode Verb operation code. This field identifies the verb.

format Identifies the format of the VCB. The value that this field must be set to in order to specify the current version of the VCB is documented individually under each verb.

primary_rc Primary return code. Possible values for each verb are listed in the following sections.

secondary_rc Secondary return code. This supplements the information provided by the primary return code.

Chapter 13. Management Services Entry Points

This chapter describes the entry points for Management Services verbs.

© Copyright IBM Corp. 1989, 1996

WinMS()

This provides a synchronous entry point for issuing the following MS API verbs:

SEND_MDS_MU
TRANSFER_MS_DATA

Syntax

void WINAPI WinMS(long vcb, unsigned short vcb_size);

Parameter	Description
vcb	Pointer to verb control block
vcb_size	Number of bytes in the verb control block

Returns

No return value. The **primary_rc** and **secondary_rc** fields in the verb control block indicate any error.

Remarks

This is the main synchronous entry point for the MS API. This call blocks until the verb completes.

WinMSCleanup()

This function terminates and deregisters an application from the MS API.

Syntax

```
BOOL WINAPI WinMSCleanup(void);
```

Returns

The return value specifies whether the deregistration was successful. If the value is not zero, the application was successfully deregistered. The application was not deregistered if a value of zero is returned.

Remarks

Use **WinMSCleanup()** to indicate deregistration of a MS application from the MS API.

WinMSCleanup unblocks any thread waiting in **WinMSGetIndication**. These return with WMSNOTREG (the application is not registered to receive indication). **WinMSCleanup** unregisters the application for all indications. **WinMSCleanup** returns any outstanding verb (synchronous or asynchronous) with the error AP_CANCELLED. However, the verb completes inside the node.

It is not a requirement to use **WinMSStartup** and **WinMSCleanup**. However, an application must be consistent in its use of these calls. You should use both of them or never use either of them. If a MS application does not call **WinMSStartup** and **WinMSCleanup**, it may not be able to operate successfully with later versions of Personal Communications.

Note: See also **WinMSStartup()**.

WinMSStartup()

This function allows an application to specify the version of MS API required and to retrieve the version of the API supported by the product. This function may be called by an application before issuing any further MS API calls to register itself.

Syntax

```
int WINAPI WinMSStartup(WORD wVersionRequired,
                        LPWMSDATA  msdata);
```

Parameter	Description
wVersionRequired	Specifies the version of MS API support required. The high-order byte specifies the minor version (revision) number; the low-order byte specifies the major version number. Returns version of MS API and description of MS implementation.

Returns

The return value specifies whether the application was registered successfully and whether the MS API implementation can support the specified version number. If the value is zero, it was registered successfully and the specified version can be supported. Otherwise, the return value is one of the following:

WMSSYSERROR
The underlying network subsystem is not ready for network communication.

WMSVERNOTSUPPORTED
The version of MS API support requested is not provided by this particular MS API implementation.

WMSBADPOINTER
Invalid msdata parameter.

Remarks

WinMSStartup is intended to help with compatibility with future versions of the API. The current version supported is 1.0.

It is not a requirement to use **WinMSStartup** and **WinMSCleanup**. However, an application must be consistent in its use of these calls. You should use both of them or never use either of them. If a MS application does not call **WinMSStartup** and **WinMSCleanup**, it may not be able to operate successfully with later versions of Personal Communications.

Note: See also **WinMSCleanup()**.

WinMSRegisterApplication()

This function registers the application as an NMVT-level application, an MDS-level application, or an alert handler. Such registrations determine which unsolicited indications the application receives.

An NMVT-level application receives NMVT_RECEIVED indications.

An MDS-level application receives MDS_MU_RECEIVED indications and also FP_NOTIFICATION indications when focal-point status changes.

An alert handler receives ALERT_INDICATION indications.

Note: It is also possible to register to receive NMVTs with conversion to MDS MUs.

Applications which do not process these indications should not call **WinMSRegisterApplication**.

Syntax

```
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);
```

Parameter	Description
reg_type	Registration type WMSNMVTAPP — NMVT-level application (or MDS-level application registering to receive NMVTs) WMSMDSAPP — MDS-level application WMSALERTHANDLER — Alert handler
ms_appl_name	MS application name. Valid names may be either an 8-byte alphanumeric type-1134 EBCDIC string, padded with trailing space (X'40') characters if necessary, or one of the MS Discipline-Specific Application Programs specified in Appendix D of *SNA Management Services Reference*, padded with trailing space (X'40') characters. This name is used when **reg_type** is WMSNMVTAPP or WMSMDSAPP. The name is not applicable when **reg_type** is WMSALERTHANDLER.
vector_key	MS major vector key(s) accepted by the application Permitted values are: X'YYYY' — specific major vector key AP_SPCF_KEYS — major vector keys X'8 61' through X'8 64' AP_ALL_KEYS — all major vector keys This key is used when **reg_type** is WMSNMVTAPP. The key is not applicable when **reg_type** is WMSMDSAPP or WMSALERTHANDLER.

mds_conv_reqd Specifies whether the registering application is MDS-level and requires NMVTs sent to it to be converted to MDS MUs

(AP_YES or AP_NO)

This parameter is used when **reg_type** is WMSNMVTAPP. The parameter is not applicable when **reg_type** is WMSMDSAPP or WMSALERTHANDLER.

ms_category Specifies an MS category when the application desires information pertaining to the focal point for that category. The MS category may be either one of the category codes specified in the MS Discipline-Specific Application Programs table of Appendix D of *SNA Management Services Reference* padded with trailing space (X'40') characters or a user-defined category. User-defined category names should be an 8-byte alphanumeric type-1134 EBCDIC string, padded with trailing space (X'40') characters if necessary.

This parameter is used when **reg_type** is WMSMDSAPP. The parameter is not applicable when **reg_type** is WMSNMVTAPP or WMSALERTHANDLER.

max_rcv_size Maximum number of bytes the application is capable of receiving in one chunk. MDS MUs bigger that this size will be segmented, and each segment delivered in a separate MDS_MU_RECEIVED indication.

This parameter is used when **reg_type** is WMSMDSAPP. The parameter is not applicable when **reg_type** is WMSNMVTAPP or WMSALERTHANDLER.

alert_dest Specifies whether the application wishes to be the only destination of all alerts. If this is set to AP_YES then all alerts will be routed to the application, and will not be routed anywhere else. If set to AP_NO, alerts will be routed to the application and over the SNA Network in the usual way.

This parameter is used when **reg_type** is WMSALERTHANDLER. The parameter is not applicable when **reg_type** is WMSNMVTAPP or WMSMDSAPP.

primary_rc Returned: primary return code

secondary_rc Returned: secondary return code

Returns

The function returns a value indicating whether the registration was successful. If the value is not zero, the registration was successful. If the value is zero, the registration was not successful.

Remarks

Applications can make multiple calls to register more than one class of indications.

Applications that call **WinMSRegisterApplication** must call **WinMSGetIndication** to receive indications that are queued for them.

Note: See also **WinMSUnregisterApplication** and **WinMSGetIndication**.

WinMSUnregisterApplication()

This function deregisters the application, reversing the effect of an earlier **WinMSRegisterApplication** call, and stopping further indications from being queued for the application.

Syntax

```
BOOL WINAPI WinMSUnregisterApplication(unsigned short reg_type,
                                       unsigned  char     ms_appl_name,
                                        unsigned short  primary_rc,
                                       unsigned  long     secondary_rc);
```

Parameter	Description
reg_type	Registration type. It can have one of the following values: WMSNMVTAPP NMVT-level application WMSMDSAPP MDS-level application WMSALERTHANDLER Alert handler
ms_appl_name	MS application name. Valid names may be either an 8-byte alphanumeric type-1134 EBCDIC string, padded with trailing space (X'40') characters if necessary, or one of the MS Discipline-Specific Application Programs specified in Appendix D of *SNA Management Services Reference*, padded with trailing space (X'40') characters. This parameter is used when **reg_type** is WMSNMVTAPP or WMSMDSAPP. The parameter is not applicable when **reg_type** is WMSALERTHANDLER.
primary_rc	Returned: primary return code
secondary_rc	Returned: secondary return code

Returns

The function returns a value indicating whether the unregistration was successful. If the value is not zero, the unregistration was successful. If the value is zero, the unregistration was not successful.

Remarks

Each call to **WinMSUnregisterApplication** terminates a registration made by an earlier call to **WinMSRegisterApplication**. An application which has made multiple calls to **WinMSRegisterApplication** needs to make multiple calls to **WinMSUnregisterApplication** in order to terminate all its registrations.

WinMSUnregisterApplication and **WinMSCleanup** differ as follows:

WinMSUnregisterApplication terminates an earlier registration to receive indications, but does not prevent the application from making other MS API calls (for example, WinMS).

WinMSCleanup terminates use of the MS API.

It is possible that indications may already be queued for an application when the application calls **WinMSUnregisterApplication** Any such indications remain queued, and the application should call **WinMSGetIndication** to receive and

process them. Once they have been unregistered, no new indications will be queued for the application.

Note: See also **WinMSRegisterApplication** and **WinMSGetIndication**.

WinMSGetIndication()

This allows the application to received unsolicited indications.

Syntax

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

Parameter	Description
buffer	Pointer to buffer into which to receive the indication.
buffer_size	Size of buffer. Returned: the size of the indication.
timeout	Time to wait for indication in milliseconds.

Returns

The function returns a value indicating whether an indication was received.

0 Indication returned.

WMSTIMEOUT
Timeout waiting for indication.

WMSSYSNOTREADY
The underlying network subsystem is not ready for network communication.

WMSNOTREG
The application is not registered to receive indications.

WMSBADSIZE
The buffer is too small to receive the indication. Reissue the **WinMSGetIndication** call with a large enough buffer. The size of the indication is returned in the **buffer_size** parameter.

WMSBADPOINTER
Either the buffer or **buffer_size** parameter is invalid.

WMSSYSERROR
An unexpected system error has occurred.

Remarks

This is a blocking call, it returns in one of the following circumstances:

An indication is returned
The timeout expires
The application issues a **WinMSCleanup** call
The product is stopped
A system error occurs

Note: See also **WinMSRegisterApplication** and **WinMSUnregisterApplication**.

WinMSGetIndication()

Chapter 14. Management Services Verbs

The Management Services API verbs by Personal Communications enable an application to send alerts and MDS MU, and to receive indications when the node receives MDS or NMVT data or issues an alert.

© Copyright IBM Corp. 1989, 1996

TRANSFER_MS_DATA

This verb is used by NMVT-level applications to send unsolicited alerts and to respond to previously-received NMVT requests.

TRANSFER_MS_DATA is also used by MDS-level applications to send unsolicited alerts. This verb can be used by the application using the WinMS call.

VCB Structure

```
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;
```

Supplied Parameters

The application supplies the following parameters:

opcode SV_TRANSFER_MS_DATA

data_type Specifies the type of data enclosed. MS processes the data as described below. Allowed values:

SV_NMVT
The data contains a complete NMVT request unit. MS converts the data to MDS_MU or CP_MSU format if the data contains an alert, and the alert is to be sent to an MDS-level or migration-level focal point. This is the type required when an application is responding to an NMVT_RECEIVED signal.

SV_ALERT_SUBVECTORS
The data contains MS subvectors in the SNA-defined format for an Alert major vector. MS adds an NMVT header and an alert major vector header. Subsequently, MS converts the data to MDS_MU or CP_MSU format if the alert is to be sent to an MDS-level or migration-level focal point.

SV_USER_DEFINED
The data contains a complete NMVT request unit. MS always logs the data, but does not send it.

SV_PDSTATS_SUBVECTORS
The data contains problem determination statistics. MS always logs the data, and if an alert handler has been registered, then MS sends it the data within an ALERT_INDICATION.

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

options Specifies optional processing on the data supplied on this verb. Note that MS processes the data primarily according to the **type** specified if there is any conflict between the **data_type** and the option specified. This parameter is a one-byte value, with individual bit settings indicating the options selected. If all options are specified, set this byte to zero.

Bit 0 is the most significant, and bit 7 is the least significant bit.

(Bits 1-3 are ignored if **data_type** is set to SV_USER_DEFINED.)

Bit 0 - Adds Date/Time (X'01') subvector to the data if set to zero.

Bit 1 - Adds Product Set ID (X'10') subvector to the data if set to zero. If the application supplies data that already contains a Product Set ID subvector, MS adds Personal Communications' Product Set ID subvector immediately before the existing one.

Bit 2 - Sends the data on an SNA session if set to zero. MS sends the data on the default SSCP-PU session if the data does not contain an alert. If the data contains an alert, MS sends the data on either an SSCP-PU session, a CP-CP session or an LU-LU session, depending on which type of session Personal Communications uses to transmit alerts to the alert focal point.

Bit 3 - Logs the data via the Personal Communications problem determination facility if set to zero.

Note: The following constants are provided in the MS header file which refer to the individual bits specified above.

SV_TIME_STAMP_SUBVECTOR
SV_PRODUCT_SET_ID_SUBVECTOR
SV_SEND_ON_SESSION
SV_LOCAL_LOGGING

Bits 4-7 - reserved.

originator_id Name of the component that issued the verb. This is an 8-byte string in a locally displayable character set. This field is only used by MS when logging the TRANSFER_MS_DATA.

pu_name Name of the physical unit to send the data to. This should be set to either an 8-byte alphanumeric type-A EBCDIC string, padded to the right with EBCDIC spaces, or set to all binary zeros if no **pu_name** is specified. Applications using TRANSFER_MS_DATA to respond to NMVT_RECEIVED signals should specify the **pu_name** received in the

NMVT_RECEIVED signal. The data contained in TRANSFER_MS_DATA signals of type SV_NMVT which do not specify a **pu_name** will be sent over the default PU session if available. TRANSFER_MS_DATA signals containing alerts should not specify a **pu_name** unless the application expressly wishes the alert data to be sent to a specific PU. This will bypass the normal MS alert routing algorithm.

dlen Length of data.

dptr Pointer to data. If this is set to NULL, then MS assumes that the data is contiguous with (and begins immediately following) the VCB.

Returned Parameters: Successful Execution

If the verb executes successfully, MS returns the following parameters.

primary_rc AP_OK

Returned Parameters: Parameter Check

If the verb fails to execute because of a parameter error, MS returns the following parameters:

primary_rc AP_PARAMETER_CHECK

secondary_rc SV_INVALID_DATA_TYPE
SV_DATA_EXCEEDS_RU_SIZE
AP_INVALID_PU_NAME

Returned Parameters: State Check

If the verb fails to execute because of a state error, MS returns the following parameters:

primary_rc AP_STATE_CHECK

secondary_rc SV_SSCP_PU_SESSION_NOT_ACTIVE

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications APPN returns the following parameters:

primary_rc AP_UNEXPECTED_SYSTEM_ERROR

MDS_MU_RECEIVED

This verb indication is sent by MS to a registered MDS-level application when:

An MDS_MU has been received from a peer MDS-level application

An NMVT has been received, and an appropriate NMVT-level application has not registered

The MDS-level application registered with a name that corresponds to the name carried within the MS major vector key in the incoming NMVT (MS performs the conversion from NMVT to MDS_MU).

VCB Structure

```
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;
```

Supplied Parameters

opcode AP_MDS_MU_RECEIVED

format Identifies the format of the VCB. This field is set to zero to specify the version of the VCB listed above.

first_message Flag indicating whether this is the first message for the MDS_MU (AP_YES or AP_NO). If the **max_rcv_size** specified in the **WinMSRegisterApplication** call is smaller than the length of the MDS_MU being delivered, the MDS_MU will be sent to the application in chunks.

last_message Flag indicating whether this is the last message for the MDS_MU (AP_YES or AP_NO).

pu_name Name of the physical unit from which the NMVT (which has been converted to an MDS_MU) originated. It is the responsibility of the application to respond to the incoming NMVT. The application uses SEND_MDS_MU to send the response. When sending responses the application must set the **pu_name** field of the SEND_MDS_MU to the **pu_name** supplied in the MDS_MU_RECEIVED signal. If the MDS_MU was received from the MDS level transport mechanism, the **pu_name** will be set to all binary zeros.

mds_mu_length Length of MDS_MU portion included with the signal.

mds_mu MDS_MU data. The data pointer is set to NULL, and the data is contiguous with (and begins immediately following) the VCB.

SEND_MDS_MU

This verb is used by a MDS-level application to send network management data other than alerts using the **WinMS** entry point. If an error occurs during the sending of the MDS_MU to the destination application, the error is reported back to the origin application in one of two ways. If the error is detected at the local node, the application will be notified via the return codes of the SEND_MDS_MU response. If the error is detected at a remote node, the error is reported by means of an error MDS_MU transported in an MDS_MU_RECEIVED VCB. MS may convert the outgoing MDS_MU to an NMVT if the destination node is to be reached via an SSCP-PU session. The application does not need to know the identity of its local node. If the application supplies 8 EBCDIC blanks in the **netid** or **nau** or both subfields of the origin location name subvector of the MDS Routing Information GDS variable, Personal Communications will supply the appropriate values. If an application does not fill in either the **netid** or **nau** but supplies fewer than 8 blanks, Personal Communications will return a secondary return code of AP_INVALID_MDS_MU_FORMAT.

VCB Structure

```
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;
```

Supplied Parameters

opcode AP_SEND_MDS_MU

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

options Specifies optional processing on the data supplied on this verb. This parameter is a one-byte value, with individual bit settings indicating the options selected. If all options are specified, set this byte to zero.

Bit 0 is the most significant, and bit 7 is the least significant bit.

Bit 0 - Adds Date/Time (X'01') subvector to the data if set to zero.

Bit 1 - Adds Product Set ID (X'10') subvector to the data if set to zero. If the application supplies data which already contains a Product Set ID subvector, then MS

adds Personal Communications' Product Set ID subvector immediately before the existing one.

Bit 2- reserved.

Bit 3 - Logs the data via the Personal Communications problem determination facility if set to zero.

Note: The following constants are provided in the MS header file that refer to bits 0, 1, and 3 specified above.

SV_TIME_STAMP_SUBVECTOR
SV_PRODUCT_SET_ID_SUBVECTOR
SV_LOCAL_LOGGING

Bit 4 - Specifies whether MS is to use default or direct routing to send the MS data to the destination application (AP_DEFAULT or AP_DIRECT).

Note: To set bit 4, use AP_DEFAULT or AP_DIRECT shifted appropriately (for example, AP_DIRECT<<3).

Bits 5-7 - reserved.

originator_id Name of component that issued the verb. This field is only used by MS when logging the SEND_MDS_MU.

pu_name Name of the physical unit to send the data to. This should be set to either an 8-byte alphanumeric type-A EBCDIC string, padded to the right with EBCDIC spaces, or set to all binary zeros if no **pu_name** is specified. Applications using SEND_MDS_MU to respond to MDS_MU_RECEIVED indications which were converted from incoming NMVTs should specify the **pu_name** received in the MDS_MU_RECEIVED signal. MDS_MUs which are to be transported using the MDS transport facility should set the **pu_name** to all binary zeros.

dlen Length of data.

dptr Pointer to data. If this is set to NULL, MS assumes that the data is contiguous with (and begins immediately following) the VCB.

Returned Parameters: Successful Execution

If the verb executes successfully, Personal Communications MS returns the following parameters:

primary_rc AP_OK

Returned Parameters: Parameter Check

If the verb fails to execute because of a parameter error, Personal Communications MS returns the following parameters:

primary_rc AP_PARAMETER_CHECK

secondary_rc	AP_INVALID_PU_NAME AP_INVALID_MDS_MU_FORMAT SV_INVALID_DATA_SIZE

Returned Parameters: State Check

If the verb fails to execute because of a state error, Personal Communications MS returns the following parameters:

primary_rc	AP_STATE_CHECK
secondary_rc	AP_SSCP_PU_SESSION_NOT_ACTIVE

Returned Parameters: System Error

If the verb does not execute because of a system error, Personal Communications APPN returns the following parameters:

primary_rc	AP_UNEXPECTED_SYSTEM_ERROR

ALERT_INDICATION

This verb indication is used by MS to send alert major vectors to a registered alert handler or registered held alert handler that will process them.

VCB Structure

```
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;
```

Supplied Parameters

opcode AP_ALERT_INDICATION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

alert_length Length of the alert data.

alert Pointer to the alert data. The data pointer is set to NULL, and the data is contiguous with (and begins immediately following) the VCB.

FP_INDICATION

If an MDS-level application has been registered for a particular MS category and the status of a focal point for that category changes, then MS sends this verb signal to the application.

VCB Structure

```
typedef struct ms_fp_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     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_INDICATION;
```

Supplied Parameters

opcode AP_FP_INDICATION

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

fp_routing Type of routing that should be specified on the SEND_MDS_MU when sending a message to the focal point (AP_DEFAULT or AP_DIRECT).

fp_data_length Length of focal point data.

fp_data Focal Point data containing a Focal Point Notification (X'E1') subvector and a Focal Point Identification (X'21') subvector. This data pointer is set to NULL, and the data is contiguous with (and begins immediately following) the VCB.

NMVT_RECEIVED

This verb signal is sent by MS to a registered NMVT-level application when an NMVT Is received from a remote node.

In routing incoming NMVTs, MS applies the following rules:

1. Try to route to an NMVT-level application registered with the major vector key carried on the incoming NMVT, else...
2. If the major vector key is one of X'8061' through X'8064', try to route to a registered NMVT-level AP_SPCF_KEYS application, else...
3. Try to route to an NMVT-level registered AP_ALL_KEYS application, else...
4. Try to route the NMVT (after conversion to an MDS_MU) to an MDS-level application, registered with the major vector key carried on the incoming NMVT, else...
5. If the major vector key is one of X'8061' through X'8064', try to route the NMVT (after conversion to an MDS_MU) to a registered MDS-level application, else...
6. Try to route (after conversion to an MDS_MU) to a registered AP_ALL_KEYS MDS-level application, else...
7. Negatively respond to the NMVT.

VCB Structure

```
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;         / Physical unit name         /
   unsigned  char      reserv3[6];      / reserved                   /
    unsigned short     nmvt_length;     / Length of incoming NMVT    /
    unsigned char      nmvt;            / NMVT data                  /
} MS_NMVT_RECEIVED;
```

Supplied Parameters

opcode AP_NMVT_RECEIVED

format Identifies the format of the VCB. Set this field to zero to specify the version of the VCB listed above.

pu_name Name of the physical unit from which the NMVT originated. It is the responsibility of the application to respond to the incoming NMVT. The application uses TRANSFER_MS_DATA to send the response. When sending responses, the application must set the **pu_name** field of the TRANSFER_MS_DATA to the **pu_name** supplied in the NMVT_RECEIVED signal.

nmvt_length Length of NMVT data.

nmvt Full NMVT, containing MS major vector of the types specified on the REGISTER_NMVT_APPLICATION. This data pointer is set to NULL, and the data is contiguous with (and begins immediately following) the VCB.

Appendix A. IBM APPN MIB Tables

The following table gives details on implementing the tables from the IBM APPN management information block (MIB), as defined by RFC1593. The table defines the following:

Node Operator Facility QUERY verb used to implement each MIB table
Input parameter settings
Any filtering operations required

(The mapping between the returned parameters and the MIB tables variables can be derived from the definition of the Node Operator Facility QUERY verbs). Personal Communications does not currently support the ibmappnNodePortDlcTraceTable and the ibmappnLsStatusTable MIB tables.

IBM MIB Table	Node Operator Facility Verb and MIB Table Variables	Input parameter settings
ibmappnNodePortTable	QUERY_PORT ibmappnNodePortName	**port_name**
ibmappnNodePortIpTable	(Note 1)	
ibmappnNodePortDlsTable	QUERY_PORT ibmappnNodePortDlsName (select entries with **dlc_type** of AP_SDLC)	**port_name**
ibmappnNodePortTrTable	QUERY_PORT ibmappnNodePortTrName (select entries with **dlc_type** of AP_TWINAX)	**port_name**
ibmappnNodeLsTable	QUERY_LS ibmappnNodeLsName	**ls_name**
ibmappnNodeLsIpTable	(Note 1)	
ibmappnNodeLsDlsTable	QUERY_LS ibmappnNodeLsDlsName (select entries with **dlc_type** of AP_SDLC)	**ls_name**
ibmappnNodeLsTrTable	QUERY_LS ibmappnNodeLsTrName (select entries with **dlc_type** of AP_TWINAX)	**ls_name**
ibmappnNnTopoRouteTable	QUERY_COS ibmappnNnTopoRouteCos	**cos_name**
ibmappnLocalTgTable	QUERY_LOCAL_TOPOLOGY ibmappnLocalTgDest ibmappnLocalTgNum	**dest** **dest_type** AP_LEARN_NODE **tg_num**
ibmappnLocalEnTable	QUERY_LOCAL_TOPOLOGY ibmappnLocalEnName (select entries with unique **dest**) (Note 2)	**dest** **dest_type** AP_END_NODE **dest_type** AP_LEARN_NODE
ibmappnLocalEnTgTable	QUERY_LOCAL_TOPOLOGY (Note 3) ibmappnLocalEnTgOrigin ibmappnLocalEnTgNum	**dest** **dest_type** AP_LEARN_NODE **tg_num**
ibmappnDirTable	QUERY_DIRECTORY_LU ibmappnDirLuName	**lu_name**

© Copyright IBM Corp. 1989, 1996

IBM MIB Table	Node Operator Facility Verb and MIB Table Variables	Input parameter settings
ibmappnCosModeTable	QUERY_MODE_TO_COS_ MAPPING ibmappnCosModeName	**mode_name**
ibmappnCosNameTable	QUERY_COS ibmappnCosName	**cos_name**

Notes:

1. Personal Communications does not support IP as a DLC type.
2. Entries with the same **dest** are ordered consecutively by QUERY_LOCAL_TOPOLOGY.
3. The ibmappnLocalEnTgTable views TGs from the attached End Node's perspective (that is, as a TG from the End Node). However, a network node compliant with the current level of the APPN architecture only stores End Node TG information for TGs between itself and directly attached End Nodes. Therefore all the entries in this table have ibmappnLocalEnTgDest set to the name of the local node (ibmappnNodeCpName).

Glossary

This glossary includes terms and definitions from:

The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

The ANSI/EIA Standard—440-A, *Fiber Optic Terminology*. Copies may be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue, N.W., Washington, DC 20006. Definitions are identified by the symbol (E) after the definition.

The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

Internet Request for Comments: 1208, *Glossary of Networking Terms*.

Internet Request for Comments: 1392, *Internet Users' Glossary*.

The *Object-Oriented Interface Design: IBM Common User Access Guidelines*, Carmel, Indiana: Que, 1992.

The following cross-references are used in this glossary:

Contrast with: This refers to a term that has an opposed or substantively different meaning.

Synonym for: This indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the glossary.

Synonymous with: This is a backward reference from a defined term to all other terms that have the same meaning.

See: This refers the reader to multiple-word terms that have the same last word.

See also: This refers the reader to terms that have a related, but not synonymous, meaning.

Deprecated term for: This indicates that the term should not be used. It refers to a preferred term, which is defined in its proper place in the glossary.

A

ACB name. (1) The label of an ACB macroinstruction used to name the ACB. (2) A name specified either on the VTAM APPL definition statement or on the VTAM application program's ACB macroinstruction. Contrast with *network name*.

accept. (1) In a VTAM application program, to establish a session with a logical unit (LU) in response to a CINIT request from a system services control point (SSCP). The session-initiation request may begin when a terminal user logs on, a VTAM application program issues a macroinstruction, or a VTAM operator issues a command. See also *acquire*. (2) An SMP process that moves distributed code and MVS-type programs to the distribution libraries.

ACCESS. In the Simple Network Management Protocol (SNMP), the clause in a Management Information Base (MIB) module that defines the minimum level of support that a managed node provides for an object.

ACF/VTAM. Advanced Communications Function for the Virtual Telecommunications Access Method. Synonym for *VTAM*.

acquire. (1) In VTAM, to take over resources that were formerly controlled by an access method in another domain or to resume control of resources that were controlled by that domain but released. Contrast with *release*. See also *resource takeover*. (2) In a VTAM application program, to initiate and establish a session with another logical unit (LU). The acquire process begins when the application program issues a macroinstruction. See also *accept*.

activate. To make a resource ready to perform its function. Contrast with *deactivate*.

active. (1) Operational. (2) Pertaining to a file, page, or program that is in main storage or memory, as opposed to a file, page, or program that must be retrieved from auxiliary storage; for example, an active page in an IBM personal computer. (3) Pertaining to a node or device that is connected or is available for connection to another node or device. (4) The state of

© Copyright IBM Corp. 1989, 1996

a resource when it has been activated and is operational. (5) In the AIX operating system, pertaining to the window pane in which the text cursor is currently positioned. (6) In VTAM, pertaining to a major or minor node that has been activated by VTAM. Most resources are activated as part of VTAM start processing or as the result of a VARY ACT command. (7) Contrast with *inactive* and *inoperative*. (8) See also *pending active session*.

active application. The application subsystem currently in an extended recovery facility (XRF) session with a terminal user. See *alternate application*.

ACTLU. Activate logical unit. In SNA, a command used to start a session on a logical unit.

ACTPU. Activate physical unit. In SNA, a command used to start a session on a physical unit.

adapter. A part that electrically or physically connects a device to a computer or to another device.

adaptive pacing. Synonym for *adaptive session-level pacing*.

adaptive session-level pacing. A form of session-level pacing in which session components exchange pacing windows that may vary in size during the course of a session. This allows transmission within a network to adapt dynamically to variations in availability and demand of buffers on a session-by-session basis. Session-level pacing occurs within independent stages along the session path according to local congestion at the intermediate and endpoint nodes. Synonymous with *adaptive pacing* and *adaptive session pacing*. Contrast with *fixed session-level pacing*.

adaptive session pacing. Synonym for *adaptive session-level pacing*.

address. In data communication, the unique code assigned to each device, workstation, or user connected to a network.

address space. (1) A set of addresses used to uniquely identify network accessible units, sessions, adjacent link stations, and links in a node for each network in which the node participates. An APPN node has one address space for intranode routing and one for each transmission group on which it can send message units. (2) In MPTN architecture, the set of all legal transport addresses that may be formed according to the rules of a given address type. These rules include the maximum number of characters that can be in the address and the permissible characters. Each transport protocol has its own set of rules. Since addresses in one protocol may also be legitimate in another protocol, MTPN qualifies all transport addresses with an address type.

adjacent link station (ALS). (1) In SNA, a link station directly connected to a given node by a link connection over which network traffic can be carried.

Note: Several secondary link stations that share a link connection do not exchange data with each other and therefore are not adjacent to each other.

(2) With respect to a specific node, a link station partner in an adjacent node.

adjacent nodes. Two nodes connected together by at least one path that connects no other node. (T)

Administrative Domain. A collection of hosts and routers, and the interconnecting networks, managed by a single administrative authority.

Advanced Peer-to-Peer Networking (APPN). An extension to SNA featuring (a) greater distributed network control that avoids critical hierarchical dependencies, thereby isolating the effects of single points of failure; (b) dynamic exchange of network topology information to foster ease of connection, reconfiguration, and adaptive route selection; (c) dynamic definition of network resources; and (d) automated resource registration and directory lookup. APPN extends the LU 6.2 peer orientation for end-user services to network control and supports multiple LU types, including LU 2, LU 3, and LU 6.2.

Advanced Peer-to-Peer Networking (APPN) end node. A node that provides a broad range of end-user services and supports sessions between its local control point (CP) and the CP in an adjacent network node. It uses these sessions to dynamically register its resources with the adjacent CP (its network node server), to send and receive directory search requests, and to obtain management services. An APPN end node can also attach to other end nodes.

Advanced Peer-to-Peer Networking (APPN) network. A collection of interconnected network nodes and their client end nodes.

Advanced Peer-to-Peer Networking (APPN) network node. A node that offers a broad range of end-user services and that can provide the following:

Distributed directory services, including registration of its domain resources to a central directory server

Topology database exchanges with other APPN network nodes, enabling network nodes throughout the network to select optimal routes for LU-LU sessions based on requested classes of service

Session services for its local LUs and client end nodes

Intermediate routing services within an APPN network

Advanced Peer-to-Peer Networking (APPN) node. An APPN network node or an APPN end node.

advanced program-to-program communication (APPC). (1) The general facility characterizing the LU 6.2 architecture and its various implementations in products. (2) Sometimes used to refer to the LU 6.2 architecture and its product implementations as a whole, or to an LU 6.2 product feature in particular, such as an APPC application programming interface.

agent. (1) In systems management, a user that, for a particular interaction, has assumed an agent role. (2) An entity that represents one or more managed objects by (a) emitting notifications regarding the objects and (b) handling requests from managers for management operations to modify or query the objects. (3) A system that assumes an agent role.

alert. (1) A message sent to a management services focal point in a network to identify a problem or an impending problem. (2) In SNA management services (SNA/MS), a high priority event that warrants immediate attention.

alert focal point. The system in a network that receives and processes (logs, displays, and optionally forwards) alerts. An alert focal point is a subset of a problem management focal point.

all points addressable (APA). In computer graphics, pertaining to the ability to address and display or not display each picture element (pel) on a display surface.

allocate. (1) To assign a resource, such as a disk or diskette file, to perform a task. (2) An LU 6.2 application programming interface (API) verb used to assign a session to a conversation for the conversation's use. (3) Contrast with *deallocate*.

Already-Verified indicator. An indication in the Attach function management header that a conversation request on a session between two logical units is being sent with a user ID, but without a password, because the ID-password confirmation has already been verified.

ALS. Adjacent link station.

alternate application. The subsystem that is prepared to take over a particular active application's extended recovery facility (XRF) sessions with terminal users in case the application fails. See *active application*.

AND operation. Synonym for *conjunction*.

ANR. Automatic network routing.

any-mode. In VTAM, the following:

The form of a RECEIVE request that obtains input from any one (unspecified) session

The form of an ACCEPT request that completes the establishment of a session by accepting any one (unspecified) queued CINIT request.

Contrast with *specific-mode*. See *continue-any mode*.

AP. Alternate printer.

APA. All points addressable.

API. Application programming interface.

APPC. Advanced program-to-program communication.

APPL. Application program.

application. A collection of software components used to perform specific types of user-oriented work on a computer.

application program. (1) A program written for or by a user that applies to the user's work, such as a program that does inventory control or payroll. (2) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

application programming interface (API). (1) The set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program. (2) In VTAM, the language structure used in control blocks so that application programs can reference them and be identified to VTAM.

apply. An SMP process that moves distributed code and MVS-type programs to the system libraries.

Apply. A push button that carries out the selected choices in a window without closing the window.

APPN. Advanced Peer-to-Peer Networking.

APPN end node. See *Advanced Peer-to-Peer Networking (APPN) end node*.

APPN network. See *Advanced Peer-to-Peer Networking (APPN) network*.

APPN network node. See *Advanced Peer-to-Peer Networking (APPN) network node*.

APPN node. See *Advanced Peer-to-Peer Networking (APPN) node*.

area. In Internet and DECnet routing protocols, a subset of a network or gateway grouped together by definition of the network administrator. Each area is self-contained; knowledge of an area's topology remains hidden from other areas.

argument. (1) An independent variable. (I) (A) (2) Any value of an independent variable; for example, a search key; a number identifying the location of an item in a table. (I) (A) (3) A parameter passed between a calling program and a called program.

ASCII (American National Standard Code for Information Interchange). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

asynchronous (ASYNC). (1) Pertaining to two or more processes that do not depend upon the occurrence of specific events such as common timing signals. (T) (2) Without regular time relationship; unexpected or unpredictable with respect to the execution of program instructions.

asynchronous operation. Simultaneous operations of software or hardware. In software, an operation, such as a request for session establishment or data transfer, in which the application program is allowed to continue execution while the operation is performed. The access method informs the application program after the operation is completed. Contrast with *synchronous operation*.

asynchronous request. In VTAM, a request for an asynchronous operation. Contrast with *synchronous request*.

asynchronous transfer mode (ATM). A transfer mode in which the information is organized into cells; it is asynchronous in the sense that the recurrence of cells containing information from an individual user is not necessarily periodic.

ATM. Asynchronous transfer mode.

attach. (1) In programming, to create a task that can be executed asynchronously with the execution of the mainline code. (2) To connect a device logically to a ring network.

automatic network routing (ANR). In High-Performance Routing (HPR), a highly efficient routing protocol that minimizes cycles and storage requirements for routing network layer packets through intermediate nodes on the route.

autotask. (1) An unattended NetView operator station task that does not require a terminal or a logged-on user. Autotasks can run independently of VTAM and are typically used for automated console operations. (2) Contrast with *logged-on operator*.

available. In VTAM, pertaining to a logical unit that is active, connected, enabled, and not at its session limit.

available time. From the point of view of a user, the time during which a functional unit can be used. (I) (A)

B

back-level. Pertaining to an earlier release of an IBM product, which may not support a particular, current function.

backup focal point. A focal point that provides management services support for a particular category for a node in the event of a communications failure with the primary focal point. Both assigned focal points (explicit and implicit) and default focal points can have backup counterparts. Contrast with *primary focal point*.

basic conversation. An LU 6.2 conversation type specified by the allocating transaction program. Transaction programs using basic conversation have available to them a wider variety of LU 6.2 functions, but they are responsible for more of their own error recovery and must manage details of the data stream used on the conversation. Contrast with *mapped conversation*.

basic transmission unit (BTU). (1) In SNA, the unit of data and control information passed between path control components. A BTU can consist of one or more path information units (PIUs). (2) See also *blocking of PIUs*.

batch. (1) An accumulation of data to be processed. (2) A group of records or data processing jobs brought together for processing or transmission. (3) Pertaining to activity involving little or no user action. Contrast with *interactive*.

behavior. (1) Ideally, a collection of assertions that describe the allowed states that a managed object can assume. An assertion can be a precondition, a postcondition, or an invariant. In practice, the behavior is often an informal description of the semantics of attributes, operations, and notifications. (2) The way in which managed objects, name bindings, attributes, notifications, and operations interact with the actual resources that they model and with each other.

Berkeley Internet Name Domain (BIND). The Berkeley implementation of the Domain Name System (DNS).

bid. In the contention form of invitation or selection, an attempt by the computer or by a station to gain control of a line in order to transmit data.

bidder. See *bidder session*.

bidder session. The half-session defined at session activation as having to request and receive permission from the other half-session to begin a bracket. Contrast with *first-speaker session*. Synonym for *contention-loser session*.

binary synchronous communication (BSC). (1) A form of telecommunication line control that uses a standard set of transmission control characters and control character sequences, for binary synchronous transmission of binary-coded data between stations. (2) Contrast with *Synchronous Data Link Control (SDLC)*.

bind. To relate an identifier to another object in a program; for example, to relate an identifier to a value, an address or another identifier, or to associate formal parameters and actual parameters. (T)

BIND. (1) In SNA, a request to activate a session between two logical units (LUs). See also *session activation request*. Contrast with *UNBIND*. (2) Berkeley Internet Name Domain.

BIND pacing. A technique by which the address space manager (ASM) at one node controls the rate of transmission of BIND requests of a sending ASM at another node. BIND pacing can be used to prevent BIND standoff, in which each of two nodes has reserved most of its resources for sessions it is attempting to initiate through the other and thus rejects any BINDs received from the other.

bit. Either of the digits 0 or 1 when used in the binary numeration system. (T)

BIU segment. In SNA, the portion of a basic information unit (BIU) that is contained within a path information unit (PIU). It consists of either (a) a request/response header (RH) followed by all or a part of a request/response unit (RU) or (b) a part of an RU. Synonymous with *segment*.

block. A string of data elements recorded or transmitted as a unit. The elements may be characters, words, or physical records. (T)

blocking of PIUs. In SNA, an optional function of path control that combines multiple path information units (PIUs) in a single basic transmission unit (BTU).

Note: When blocking is not done, a BTU consists of one PIU.

bracket protocol. In SNA, a data flow control protocol in which exchanges between two session partners are achieved through the use of brackets, with one partner designated at session activation as the first speaker and the other as the bidder. The bracket protocol involves bracket initiation and termination rules.

bridge. (1) A functional unit that interconnects two local area networks that use the same logical link control protocol but may use different medium access control protocols. (T) (2) A functional unit that interconnects multiple LANs (locally or remotely) that use the same logical link control protocol but that can use different medium access control protocols. A bridge forwards a frame to another bridge based on the medium access control (MAC) address. (3) In the connection of local loops, channels, or rings, the equipment and techniques used to match circuits and to facilitate accurate data transmission. (4) Contrast with *gateway* and *router*.

broadcast. (1) Transmission of the same data to all destinations. (T) (2) Simultaneous transmission of data to more than one destination. (3) Contrast with *multicast*.

broadcast Locate search. Synonym for *broadcast search*.

broadcast search. The simultaneous propagation of a search request to all network nodes in an APPN network. This type of search may be used when the location of a resource is unknown to the requester. Contrast with *directed search*. Synonymous with *broadcast Locate search*.

BSC. Binary synchronous communication.

BTU. Basic transmission unit.

buffer. (1) A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another. (A) (2) To allocate and schedule the use of buffers. (A) (3) A portion of storage used to hold input or output data temporarily.

bypass. To eliminate a station or an access unit from a ring network by allowing the data to flow in a path around it.

byte. (1) A string that consists of a number of bits, treated as a unit, and representing a character. (T) (2) A binary character operated upon as a unit and usually shorter than a computer word. (A) (3) A group of 8 adjacent binary digits that represent one EBCDIC character.

C

C language. A language used to develop software applications in compact, efficient code that can be run on different types of computers with minimal change.

cache. (1) A special-purpose buffer storage, smaller and faster than main storage, used to hold a copy of instructions and data obtained from main storage and

likely to be needed next by the processor. (T) (2) A buffer storage that contains frequently accessed instructions and data; it is used to reduce access time. (3) An optional part of the directory database in network nodes where frequently used directory information may be stored to speed directory searches. (4) To place, hide, or store in a cache.

call. (1) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (I) (A) (2) In data communication, the actions necessary to make a connection between two stations on a switched line. (3) In communications, a conversation between two users. (4) To transfer control to a procedure, program, routine, or subroutine. (5) To attempt to contact a user, regardless of whether the attempt is successful.

calling. (1) The process of transmitting selection signals in order to establish a connection between data stations. (I) (A) (2) In X.25 communications, pertaining to the location or user that makes a call.

Cancel. A push button that removes a window without applying any changes made in that window.

card. (1) An electronic circuit board that is plugged into a slot in a system unit. (2) A plug-in circuit assembly. (3) See also *adapter*. (4) In NetView for AIX, see *event card*.

case-sensitive. Pertaining to the ability to distinguish between uppercase and lowercase letters.

CD. Compact disc.

CD-ROM. High-capacity read-only memory in the form of an optically read compact disc.

CDS. (1) Control data set. (2) Configuration data set. (3) Central directory server.

central directory. A repository for storing resource location information centrally registered by network nodes or cached as the result of network searches.

central directory server (CDS). A network node that provides a repository for information on network resource locations; it also reduces the number of network searches by providing a focal point for queries and broadcast searches and by caching the results of network searches to avoid later broadcasts for the same information.

chain. (1) A group of logically linked user data records processed by LU 6.2. (2) A group of request units delimited by begin-chain and end-chain. Responses are always single-unit chains. See *RU chain*.

channel. (1) A path along which signals can be sent, for example, data channel, output channel. (A) (2) In data communication, a means of one-way transmission. (3) A functional unit, controlled by the processor, that handles the transfer of data between processor storage and local peripheral equipment.

channel-attached. (1) Pertaining to the attachment of devices directly by input/output channels to a host processor. (2) Pertaining to devices attached to a controlling unit by cables, rather than by telecommunication lines. (3) Contrast with *link-attached*. (4) Synonymous with *local*.

child. Pertaining to a secured resource, either a file or library, that uses the user list of a parent resource. A child resource can have only one parent resource. A child is a process, started by a parent process, that shares the resources of the parent process. Contrast with *parent*.

CICS. Customer Information Control System.

circuit. (1) One or more conductors through which an electric current can flow. See *physical circuit* and *virtual circuit*. (2) A logic device.

class. (1) In object-oriented design or programming, a group of objects that share a common definition and that therefore share common properties, operations, and behavior. Members of the group are called instances of the class. (2) In the AIX operating system, pertaining to the I/O characteristics of a device. System devices are classified as block or character devices.

class of service (COS). A set of characteristics (such as route security, transmission priority, and bandwidth) used to construct a route between session partners. The class of service is derived from a mode name specified by the initiator of a session.

cleanup. In SNA products, a network services request, sent by a system services control point (SSCP) to a logical unit (LU), that causes a particular LU-LU session with that LU to be ended immediately without requiring the participation of either the other LU or its SSCP.

client. (1) A functional unit that receives shared services from a server. (T) (2) A user. (3) In an AIX distributed file system environment, a system that is dependent on a server to provide it with programs or access to programs. (4) Synonymous with *requester*.

client/server. In communications, the model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called a client; the answering program is called a server.

CNOS. Change number of sessions.

code page. (1) An assignment of graphic characters and control function meanings to all code points; for example, assignment of characters and meanings to 256 code points for an 8-bit code, assignment of characters and meanings to 128 code points for a 7-bit code. (2) In the Print Management Facility, a font library member that associates code points and character identifiers. A code page also identifies invalid code points. (3) A particular assignment of hexadecimal identifiers to graphic characters. (4) In AFP support, a font file that associates code points and graphic character identifiers.

command. (1) A request from a terminal for the performance of an operation or the execution of a particular program. (2) In SNA, any field set in the transmission header (TH), request header (RH), and sometimes portions of a request unit (RU), that initiates an action or that begins a protocol; for example: (a) Bind Session (session-control request unit), a command that activates an LU-LU session, (b) the change-direction indicator in the RH of the last RU of a chain, (c) the virtual route reset window indicator in an FID4 transmission header. (3) See also *VTAM operator command*.

command prompt. A displayed character or string of characters that indicates that a user may enter a command to be processed.

common operations services (COS). The portion of SNA management services that pertains to the major vectors for limited remote operations control.

Common Programming Interface for Communications (CPI-C). An evolving application programming interface (API), embracing functions to meet the growing demands from different application environments and to achieve openness as an industry standard for communications programming. CPI-C provides access to interprogram services such as (a) sending and receiving data, (b) synchronizing processing between programs, and (c) notifying a partner of errors in the communication.

communication management configuration host node. The type 5 host processor in a communication management configuration that does all network-control functions in the network except for the control of devices channel-attached to data hosts. Synonymous with *communication management host*. Contrast with *data host node*.

communication management host. Synonym for *communication management configuration host node*. Contrast with *data host*.

Communications Manager/2. An IBM licensed program that supports the development and use of OS/2 applications involving two or more connected systems or workstations. Communications Manager/2 provides multiple concurrent connectivities using different protocols for IBM 3270 and 5250 emulation sessions, printer sessions, and file transfers. It supports a range of application programming interfaces (APIs), which may be called concurrently and are designed for a variety of applications. Communications Manager/2 includes the necessary interfaces for network management.

compact disc (CD). (1) A disc, usually 4.75 inches in diameter, from which data is read optically by means of a laser. (2) A disc with information stored in the form of pits along a spiral track. The information is decoded by a compact-disc player and interpreted as digital audio data, which most computers can process.

compile. (1) To translate all or part of a program expressed in a high-level language into a computer program expressed in an intermediate language, an assembly language, or a machine language. (T) (2) To prepare a machine language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one computer instruction for each symbolic statement, or both, as well as performing the function of an assembler. (A) (3) To translate a source program into an executable program (an object program). (4) To translate a program written in a high-level programming language into a machine language program.

compiler. (1) A program that translates a source program into an executable program (an object program). (2) A program that decodes instructions written as pseudo codes and produces a machine language program to be executed at a later time.

component. (1) Hardware or software that is part of a functional unit. (2) A part of a structured type or value, such as an array element or a record field.

computer. A functional unit that can perform substantial computations, including numerous arithmetic operations and logic operations without human intervention during a run. In information processing, the term computer usually describes a digital computer. A computer may consist of a stand-alone unit or may consist of several interconnected units. (T)

configuration. (1) The manner in which the hardware and software of an information processing system are organized and interconnected. (T) (2) The devices and programs that make up a system, subsystem, or network. (3) In CCP, the arrangement of controllers, lines, and terminals attached to an IBM 3710 Network Controller. Also, the collective set of item definitions that describe such a configuration.

configuration management. The control of information necessary to identify both physical and logical information system resources and their relationship to one another.

configure. To describe to a system the devices, optional features, and programs installed on the system.

congestion. See *network congestion*.

conjunction. The Boolean operation whose result has the Boolean value 1 if and only if each operand has the Boolean value 1. (I) (A) Synonymous with *AND operation*.

connected. In VTAM, the state of a physical unit (PU) or a logical unit (LU) that has an active physical path to the host processor containing the system services control point (SSCP) that controls the respective PU or LU.

connection. (1) In data communication, an association established between functional units for conveying information. (I) (A) (2) In Open Systems Interconnection architecture, an association established by a given layer between two or more entities of the next higher layer for the purpose of data transfer. (T) (3) In SNA, the network path that links together two logical units (LUs) in different nodes to enable them to establish communications. (4) In X.25 communication, a virtual circuit between two data terminal equipments (DTEs). A switched virtual circuit (SVC) connection lasts for the duration of a call; a permanent virtual circuit (PVC) is a permanent connection between the DTEs. (5) In TCP/IP, the path between two protocol applications that provides reliable data stream delivery service. In the Internet, a connection extends from a TCP application on one system to a TCP application on another system. (6) In system communications, a line over which data can be passed between two systems or between a system and a device. (7) Synonym for *physical connection*.

connection network. (1) A representation within an APPN network of a shared-access transport facility (SATF), such as a token ring, that allows nodes identifying their connectivity to the SATF by a common virtual routing node to communicate without having individually defined connections to one another. (2) Synonymous with *link connection network*.

connectivity. (1) The capability of a system or device to be attached to other systems or devices without modification. (T) (2) The capability to attach a variety of functional units without modifying them.

constraints. In NetDA/2, the set of essential requirements specified with the node, connection, or application definitions. A change in a constraint value changes the input to the network design. Contrast with *parameters*.

contention. In a session, a situation in which both NAUs attempt to initiate the same action at the same time, such as when both attempt to send data in a half-duplex protocol (half-duplex contention), or both attempt to start a bracket (bracket contention). At session initiation, one NAU is defined to be the contention winner; its action will take precedence when contention occurs. The contention loser must get explicit or implicit permission from the contention winner to begin its action.

contention-loser session. (1) To an NAU, a session for which it was defined during session initiation to be the contention loser. (2) Synonymous with *bidder session*.

contention polarity. The role of each LU when contention occurs for use of a session. One LU is the contention winner and the other LU is the contention loser.

contention-winner session. (1) To an NAU, a session for which it was defined during session initiation to be the contention winner. (2) Synonymous with *first-speaker session*.

continue-any mode. In VTAM, the state of a session or conversation that allows its input to satisfy a RECEIVE request issued in any-mode. While this state exists, input on the session or conversation can also satisfy RECEIVE requests issued in specific-mode. For conversations, continue-any mode is further qualified as either buffer continue-any or logical record continue-any. This specifies whether VTAM is to receive the data in terms of logical records or buffers. Contrast with *continue-specific mode*.

continue-specific mode. In VTAM, the state of a session or conversation that allows its input to satisfy only RECEIVE requests issued in specific-mode. Contrast with *continue-any mode*.

control block. (1) A storage area used by a computer program to hold control information. (I) (2) In the IBM Token-Ring Network, a specifically formatted block of information provided from the application program to the Adapter Support Interface to request an operation.

control data set (CDS). In NPM, an SMP data set used in the NPM installation process.

control point (CP). (1) A component of an APPN or LEN node that manages the resources of that node. In an APPN node, the CP is capable of engaging in CP-CP sessions with other APPN nodes. In an APPN network node, the CP also provides services to adjacent end nodes in the APPN network. (2) A

component of a node that manages resources of that node and optionally provides services to other nodes in the network. Examples are a system services control point (SSCP) in a type 5 subarea node, a network node control point (NNCP) in an APPN network node, and an end node control point (ENCP) in an APPN or LEN end node. An SSCP and an NNCP can provide services to other nodes.

control point management services unit (CP-MSU). The message unit that contains management services data and flows between management services function sets. This message unit is in general data stream (GDS) format. See also *management services unit (MSU)* and *network management vector transport (NMVT)*.

Control Program (CP). In VM/ESA, a component that manages the resources of a single computer so multiple computing systems appear to exist. Each of these apparent systems, or virtual machines, is the functional equivalent of an IBM System/370, 370-XA, or ESA computer.

control vector. One of a general class of RU substructures that has variable length, is carried within some enclosing structure, and has a one-byte key used as an identifier.

controller. A device that coordinates and controls the operation of one or more input/output devices, such as workstations, and synchronizes the operation of such devices with the operation of the system as a whole.

conversation. A logical connection between two transaction programs using an LU 6.2 session. Conversations are delimited by brackets to gain exclusive use of a session.

conversation-level security. See *session-level security*. See also *end-user verification*.

coordinated universal time (UTC). The time scale, based on the Système International (SI) second, as defined and recommended by the Comité Consultatif International de la Radio (CCIR) and maintained (using an atomic clock) by the Bureau International des Poids et Mesures (BIPM).

Note: The Système International is based on three fundamental units of measure—the meter, the kilogram, and the second—and is sometimes called the “MKS system” because of these units.

For most practical purposes, coordinated universal time (UTC) is equivalent to the mean solar time at the prime meridian (0 degrees longitude) of Greenwich, England, known as *Greenwich mean time (GMT)*. Synonymous with *Z time* and *Zulu time*.

Copy. A choice that places a copy of a selected object onto the clipboard.

correlator. Information that identifies a relation among things. An example is a variable field of a response that identifies the corresponding request.

COS. (1) Class of service. (2) Common operations services.

CP. (1) Control point. (2) In VM, Control Program.

CP-CP sessions. The parallel sessions between two control points, using LU 6.2 protocols and a mode name of CPSVCMG, on which network services requests and replies are exchanged. Each CP of a given pair has one contention-winner session and one contention-loser session with the other.

CP-MSU. Control point management services unit.

CP name. A network-qualified name of a control point (CP), consisting of a network ID qualifier identifying the network (or name space) to which the CP's node belongs, and a unique name within the scope of that network ID identifying the CP. Each APPN or LEN node has one CP name, assigned to it at system-definition time.

CPI-C. Common Programming Interface for Communications.

cross-domain. In SNA, pertaining to control or resources involving more than one domain.

Customer Information Control System (CICS). An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

D

DACTLU. Deactivate logical unit.

DACTPU. Deactivate physical unit.

data. (1) A re-interpretable representation of information in a formalized manner suitable for communication, interpretation, or processing. Operations can be performed upon data by humans or by automatic means. (T) (2) Any representations such as characters or analog quantities to which meaning is or might be assigned. (A) (3) A representation of facts or instructions in a form suitable for communication, interpretation, or processing by human or automatic means. Data include constants, variables, arrays, and character strings.

Note: Programmers make a distinction between instructions and the data they operate on;

however, in the usual sense of the word, data includes programs and program instructions.

data circuit. (1) A pair of associated transmit and receive channels that provide a means of two-way data communication. (I) (2) In SNA, synonym for *link connection*. (3) See also *physical circuit* and *virtual circuit*.

Notes:

1. Between data switching exchanges, the data circuit may include data circuit-terminating equipment (DCE), depending on the type of interface used at the data switching exchange.
2. Between a data station and a data switching exchange or data concentrator, the data circuit includes the data circuit-terminating equipment at the data station end, and may include equipment similar to a DCE at the data switching exchange or data concentrator location.

data flow control (DFC). In SNA, a request/response unit (RU) category used for requests and responses exchanged between the data flow control layer in one half-session and the data flow control layer in the session partner.

data host. Synonym for *data host node*. Contrast with *communication management configuration host*.

data host node. In a communication management configuration, a type 5 host node that is dedicated to processing applications and does not control network resources, except for its channel-attached or communication adapter-attached devices. Synonymous with *data host*. Contrast with *communication management configuration host node*.

data link. In SNA, synonym for *link*.

data link control (DLC). A set of rules used by nodes on a data link (such as an SDLC link or a token ring) to accomplish an orderly exchange of information.

data link control (DLC) layer. In SNA, the layer that consists of the link stations that schedule data transfer over a link between two nodes and perform error control for the link. Examples of data link control are SDLC for serial-by-bit link connection and data link control for the System/370 channel.

Note: The DLC layer is usually independent of the physical transport mechanism and ensures the integrity of data that reaches the higher layers.

data link level. (1) In the hierarchical structure of a data station, the conceptual level of control or processing logic between high level logic and the data link that maintains control of the data link. The data link level performs such functions as inserting transmit bits and deleting receive bits; interpreting address and control fields; generating, transmitting, and interpreting commands and responses; and computing and interpreting frame check sequences. See also *packet level* and *physical level*. (2) In X.25 communications, synonym for *frame level*.

data set. (1) Synonym for *file*. (2) Deprecated term for *modem*.

data types. In the NetView program, a description of the organization of panels. Data types are alerts, events, and statistics. Data types are combined with resource types and display types to describe the NetView program's display organization. See also *display types* and *resource types*.

database. (1) A collection of data with a given structure for accepting, storing, and providing, on demand, data for multiple users. (T) (2) A collection of interrelated data organized according to a database schema to serve one or more applications. (T) (3) A collection of data fundamental to a system. (A) (4) A collection of data fundamental to an enterprise. (A)

DBCS. Double-byte character set.

deactivate. To take a resource of a node out of service, rendering it inoperable, or to place it in a state in which it cannot perform the functions for which it was designed. Contrast with *activate*.

deallocate. An LU 6.2 application programming interface (API) verb that terminates a conversation, thereby freeing the session for a future conversation. Contrast with *allocate*.

default. Pertaining to an attribute, condition, value, or option that is assumed when none is explicitly specified. (I)

definite response (DR). In SNA, a protocol requested in the form-of-response-requested field of the request header that directs the receiver of the request to return a response unconditionally, whether positive or negative, to that request chain. Contrast with *exception response* and *no response*.

definition statement. (1) In VTAM, the statement that describes an element of the network. (2) In NCP, a type of instruction that defines a resource to the NCP. (3) See also *macroinstruction*. (4) See Figure 1 on page 415, Figure 2 on page 415, and Figure 3 on page 415.

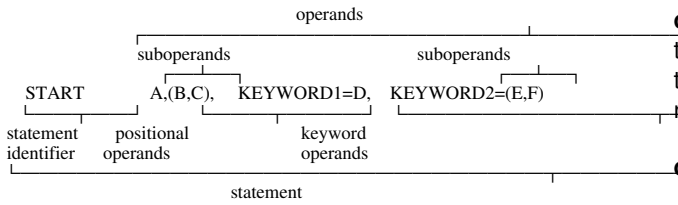

Figure 1. Example of a Language Statement

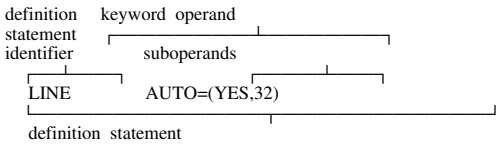

Figure 2. Example of an NCP Definition Statement

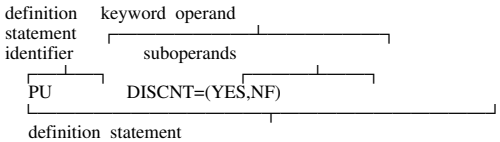

Figure 3. Example of a VTAM Definition Statement

DEL. The delete character. (A)

Delete. A choice that removes a selected object. The space it occupied is usually filled by the remaining object or objects in the window.

dependent LU. See *SSCP-dependent LU*.

dependent LU requester (DLUR). An APPN end node or an APPN network node that owns dependent LUs, but requests that a dependent LU server provide the SSCP services for those dependent LUs.

dependent LU server (DLUS). An APPN network node that provides SSCP services for a dependent LU in its own or another APPN network. Contrast with *dependent LU requester*.

destination. (1) Any point or location, such as a node, station, or a particular terminal, to which information is to be sent. (2) In ACF/TCAM, the place to which a message being handled by a message handler is to be sent. (3) An external logical unit (LU) or application program to which messages or other data are directed.

destination address. A code that identifies the location to which information is to be sent.

destination node. The node to which a request or data is sent.

DET. Device entry table.

direct routing. In Internet communications, the transmission of an Internet Protocol (IP) datagram when the destination and the source reside on the same IP network or IP subnet.

directed Locate search. Synonym for *directed search*.

directed search. A search request sent to a specific destination node known to contain a resource, such as a logical unit, to verify the continued presence of the resource at the destination node and to obtain the node's connectivity information for route calculation. Contrast with *broadcast search*. Synonymous with *directed Locate search*.

directory. (1) A table of identifiers and references to the corresponding items of data. (I) (A) (2) A database in an APPN node that lists names of resources (in particular, logical units) and records the CP name of the node where each resource is located. See *distributed directory database* and *local directory database*. (3) A type of file containing the names and controlling information for other files or other directories. (4) A listing of the files stored on a disk or diskette. (5) In VM, a Control Program (CP) disk file that defines each virtual machine's typical configuration: the user ID, password, regular and maximum allowable virtual storage, CP command privilege class or classes allowed, dispatching priority, logical editing symbols to be used, account number, and CP options desired.

directory service (DS). An application service element that translates the symbolic names used by application processes into the complete network addresses used in an OSI environment. (T)

directory services (DS). A control point component of an APPN node that maintains knowledge of the location of network resources.

disable. To make nonfunctional.

disabled. (1) Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. (2) Pertaining to the state in which a transmission control unit or audio response unit cannot accept incoming calls on a line. (3) In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is temporarily not ready to establish LU-LU sessions. An initiate request for a session with a disabled logical unit (LU) can specify that the session be queued by the SSCP until the LU becomes enabled. The LU can separately indicate whether this applies to its ability to act as a primary logical unit (PLU) or a secondary logical unit (SLU). See also *enabled* and *inhibited*.

discovery. In data communication, the automatic detection of network topology changes (for example, new and deleted nodes or new and deleted interfaces).

diskette. (1) A small magnetic disk enclosed in a jacket. (T) (2) A thin, flexible magnetic disk and a semi-rigid protective jacket, in which the disk is permanently enclosed.

display. (1) A visual presentation of data. (I) (A) (2) To present data visually. (I) (A) (3) Deprecated term for *panel*.

display levels. Synonym for *display types*.

display types. In the NetView program, a concept to describe the organization of panels. Display types are defined as total, most recent, user action, and detail. Display types are combined with resource types and data types to describe NetView's panel organization. See also *data types* and *resource types*. Synonymous with *display levels*.

distributed directory database. The complete listing of all the resources in the network as maintained in the individual directories scattered throughout an APPN network. Each node has a piece of the complete directory, but it is not necessary for any one node to have the entire list. Entries are created, modified, and deleted through system definition, operator action, automatic registration, and ongoing network search procedures. Synonymous with *distributed network directory* and *network directory database*.

distributed network directory. Synonym for *distributed directory database*.

DLC. Data link control.

DLL. Dynamic link library.

DLUR. Dependent LU requester.

DLUS. Dependent LU server.

domain. (1) That part of a computer network in which the data processing resources are under common control. (T) (2) In SNA, see *end node domain*, *network node domain*, and *system services control point (SSCP) domain*. (3) In Open Systems Interconnection (OSI), a part of a distributed system or a set of managed objects to which a common policy applies. (4) In a database, all the possible values of an attribute or a data element. (5) See *Administrative Domain* and *domain name*.

domain name. In the Internet suite of protocols, a name of a host system. A domain name consists of a sequence of subnames separated by a delimiter character. For example, if the fully qualified domain name (FQDN) of a host system is ralvm7.vnet.ibm.com, each of the following is a domain name:

ralvm7.vnet.ibm.com
vnet.ibm.com
ibm.com

domain operator. In a multiple-domain network, the person or program that controls operation of resources controlled by one system services control point (SSCP). See also *network operator*.

double-byte character set (DBCS). A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with *single-byte character set (SBCS)*.

downstream. In the direction of data flow from the host to the user. Contrast with *upstream*.

drain. To honor pending allocation requests before deactivating sessions with a partner logical unit. This applies to LU 6.2 only.

DS. (1) Desired state. (2) Directory service. (3) Directory services.

dynamic. (1) In programming languages, pertaining to properties that can only be established during the execution of a program; for example, the length of a variable-length data object is dynamic. (I) (2) Pertaining to an operation that occurs at the time it is needed rather than at a predetermined or fixed time. (3) Contrast with *static*.

dynamic link library (DLL). A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a dynamic link library can be shared by several applications simultaneously.

E

EBCDIC. Extended binary-coded decimal interchange code. A coded character set of 256 8-bit characters.

echo. (1) In computer graphics, the immediate notification of the current values provided by an input device to the operator at the display console. (I) (A) (2) In word processing, to print or display each character or line as it is keyed in. (3) In data communication, a reflected signal on a communications channel. For example, on a communications terminal, each signal is displayed twice, once when entered at the local terminal and again when returned over the communications link. This allows the signals to be checked for accuracy.

Emulation Program (EP). (1) An IBM control program that allows a channel-attached IBM communication

controller to emulate the functions of an IBM 2701 Data Adapter Unit, an IBM 2702 Transmission Control, or an IBM 2703 Transmission Control. (2) See also *network control program*.

EN. End node.

enable. To make functional.

enabled. (1) Pertaining to a state of the processing unit that allows the occurrence of certain types of interruptions. (2) Pertaining to the state in which a transmission control unit or an audio response unit can accept incoming calls on a line. (3) In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is ready to establish LU-LU sessions. The LU can separately indicate whether this prevents it from acting as a primary logical unit (PLU) or a secondary logical unit (SLU). See also *disabled* and *inhibited*.

ENCP. End-node control point.

encryption. In computer security, the process of transforming data into an unintelligible form in such a way that the original data either cannot be obtained or can be obtained only by using a decryption process.

end node (EN). (1) See *Advanced Peer-to-Peer Networking (APPN) end node* and *low-entry networking (LEN) end node*. (2) In communications, a node that is frequently attached to a single data link and cannot perform intermediate routing functions.

end node domain. An end node control point, its attached links, and its local LUs.

end-user verification. For logical unit (LU) 6.2, checking the identification of users by means of identifiers and passwords on attach function-management headers (FMHs). See *partner-LU verification*. See also *conversation-level security*.

entry point (EP). (1) The address or label of the first instruction executed on entering a computer program, routine, or subroutine. A computer program, routine, or subroutine may have a number of different entry points, each perhaps corresponding to a different function or purpose. (I) (A) (2) In SNA, a type 2.0, type 2.1, type 4, or type 5 node that provides distributed network management support. It sends network management data about itself and the resources it controls to a focal point for centralized processing, and it receives and executes focal-point initiated commands to manage and control its resources.

environment variable. A variable that specifies (a) how an operating system or another program will run or (b) the devices that the operating system will recognize.

EP. (1) Emulation Program. (2) Entry point.

ER. (1) Explicit route. (2) Exception response.

ERP. Error recovery procedures.

error. A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. (I) (A)

error recovery procedures (ERP). (1) Procedures designed to help isolate and, where possible, to recover from errors in equipment. The procedures are often used in conjunction with programs that record information on machine malfunctions. (2) A set of routines that attempt to recover from transmission errors.

Ethernet. A 10-Mbps baseband local area network that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and delayed retransmission. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD).

event. (1) An occurrence of significance to a task; for example, an SNMP trap, the opening of a window or a submap, or the completion of an asynchronous operation. (2) In the NetView and NETCENTER programs, a record indicating irregularities of operation in physical elements of a network. (3) See also *event report*.

event card. In NetView for AIX, a graphical representation, resembling a card, of the information contained in an event sent by an agent to a manager reflecting a change in the status of one of the agent's managed nodes.

event report. The unsolicited report that an event has occurred. When a managed object emits a notification, the agent uses one or more event forwarding discriminators (EFDs) to find the destinations to which the report is sent.

exception. An abnormal condition such as an I/O error encountered in processing a data set or a file.

exception response (ER). In SNA, a protocol requested in the form-of-response-requested field of a request header that directs the receiver to return a response only if the request is unacceptable as received or cannot be processed; that is, a negative response, but not a positive response, can be returned. Contrast with *definite response* and *no response*.

exchange identification (XID). A specific type of basic link unit that is used to convey node and link characteristics between adjacent nodes. XIDs are exchanged between link stations before and during link activation to establish and negotiate link and node

characteristics, and after link activation to communicate changes in these characteristics.

execute. To perform the actions specified by a program or a portion of a program. (T)

expedited flow. In SNA, a data flow designated in the transmission header (TH) that is used to carry network control, session control, and various data flow control request/response units (RUs); the expedited flow is separate from the normal flow (which carries primarily end-user data) and can be used for commands that affect the normal flow. Contrast with *normal flow*.

Note: The normal and expedited flows move in both the primary-to-secondary and secondary-to-primary directions. Requests and responses on a given flow, whether normal or expedited, usually are processed sequentially within the path, but the expedited flow traffic may be moved ahead of the normal-flow traffic within the path at queuing points in the half-sessions and for half-session support in boundary functions.

explicit focal point. An assigned focal point for which the set of nodes to be included in its sphere of control is defined locally. An explicit focal point initiates the management services capabilities exchange that brings a node into its sphere of control. Contrast with *implicit focal point*.

explicit route (ER). In SNA, a series of one or more transmission groups that connect two subarea nodes. An explicit route is identified by an origin subarea address, a destination subarea address, an explicit route number, and a reverse explicit route number. Contrast with *virtual route (VR)*.

E1. See *T1*.

F

fault. An accidental condition that causes a functional unit to fail to perform its required function. (I) (A)

feature. A part of an IBM product that may be ordered separately by the customer.

field. (1) An identifiable area in a window. Examples of fields are: an entry field, into which a user can type or place text, and a field of radio button choices, from which a user can select one choice. (2) In NetView for AIX, the building block of which objects are composed. A field is characterized by a field name, a data type (integer, Boolean, character string, or enumerated value), and a set of flags that describe how the field is treated by NetView for AIX. A field can contain data only when it is associated with an object.

file. A named set of records stored or processed as a unit. (T) Synonymous with *data set*.

filter. (1) A device or program that separates data, signals, or material in accordance with specified criteria. (A) (2) In the NetView program, a function that limits the data that is to be recorded on the database and displayed at the terminal. See also *recording filter* and *viewing filter*. (3) In the AIX operating system, a command that reads standard input data, modifies the data, and sends it to the display screen. (4) See also *IP filter*.

first speaker. See *first-speaker session*.

first-speaker session. The half-session defined at session activation as (a) able to begin a bracket without requesting permission from the other half-session to do so and (b) winning contention if both half-sessions attempt to begin a bracket simultaneously. Synonym for *contention-winner session*. Contrast with *bidder session*.

fixed pacing. Synonym for *fixed session-level pacing*.

fixed session-level pacing. A form of session-level pacing in which the data transfer rate is controlled using fixed pacing-window sizes, which are initialized at session-activation time. Synonymous with *fixed pacing*. Contrast with *adaptive session-level pacing*.

flag. (1) To mark an information item for selection for further processing. (T) (2) A character that signals the occurrence of some condition, such as the end of a word. (A)

flag (F) sequence. A unique sequence of eight bits (01111110) used to delimit the opening and closing of an SDLC frame.

flow. In NetDA/2, the amount of traffic that can pass through a node, connection, or route in both directions during a given period of time.

flow control. (1) In data communication, control of the data transfer rate. (I) (2) In SNA, the process of managing the rate at which data traffic passes between components of the network. The purpose of flow control is to optimize the rate of flow of message units with minimum congestion in the network; that is, to neither overflow the buffers at the receiver or at intermediate routing nodes, nor leave the receiver waiting for more message units. (3) See also *pacing*.

FMD. Function management data.

FNA. Free network address.

focal point (FP). (1) A system that provides centralized management services. (2) See *management services focal point (MSFP)*.

foreign host. Synonym for *remote host*.

FP. Focal point.

FQDN. Fully qualified domain name.

FQPCID. Fully qualified procedure correlator identifier.

frame. (1) In Open Systems Interconnection architecture, a data structure pertaining to a particular area of knowledge and consisting of slots that can accept the values of specific attributes and from which inferences can be drawn by appropriate procedural attachments. (T) (2) The unit of transmission in some local area networks, including the IBM Token-Ring Network. It includes delimiters, control characters, information, and checking characters. (3) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures.

frame level. Synonymous with *data link level*.

fully qualified domain name (FQDN). In the Internet suite of protocols, the name of a host system that includes all of the subnames of the domain name. An example of a fully qualified domain name is ralvm7.vnet.ibm.com. See also *host name*.

fully qualified name. (1) In SNA, synonym for *network-qualified name*. (2) In the Internet suite of protocols, see *fully qualified domain name (FQDN)*.

fully qualified procedure correlator identifier (FQPCID). A network-unique identifier that is used for the following:

- Correlating messages sent between nodes, such as correlating a Locate search request with its replies
- Identifying a session for problem determination and resolution
- Identifying a session for accounting, auditing, and performance monitoring purposes

This identifier is normally assigned at the node that contains the LU for which a procedure or session is initiated, except when that node is an end node, in which case its network node server may assign it. The FQPCID consists of a fixed-length correlator concatenated with the network-qualified name of the control point that generated the correlator.

function call. An expression that moves the path of execution from the current function to a specified function and evaluates to the return value provided by the called function. A function call contains the name of the function to which control moves and a parenthesized list of values.

function management data (FMD). An RU category used for end-user data exchanged between logical units (LUs) and for requests and responses exchanged between network services components of LUs, PUs, and control points.

G

gateway. (1) A functional unit that interconnects two computer networks with different network architectures. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures. (T) (2) A functional unit that connects two networks or subnetworks having different characteristics, such as different protocols or different policies concerning security or transmission priority. (3) The combination of machines and programs that provide address translation, name translation, and system services control point (SSCP) rerouting between independent SNA networks to allow those networks to communicate. A gateway consists of one gateway NCP and at least one gateway VTAM. (4) In the IBM Token-Ring Network, a device and its associated software that connect a local area network to another local area network or a host that uses different logical link protocols. (5) In the AIX operating system, an entity that operates above the link layer and translates, when required, the interface and protocol used by one network into those used by another distinct network. (6) In TCP/IP, synonym for *router*.

gateway NCP. (1) An NCP that performs address translation to allow cross-network session traffic. The gateway NCP connects two or more independent SNA networks. (2) Synonymous with *gateway node*.

gateway node. Synonym for *gateway NCP*.

GDS. General data stream.

general data stream (GDS). The data stream used for conversations in LU 6.2 sessions.

general data stream (GDS) variable. A type of RU substructure that is preceded by an identifier and a length field and includes either application data, user control data, or SNA-defined control data.

generation. The process of assembling and link editing definition statements so that resources can be identified to all the necessary programs in a network.

generic unbind. Synonym for *session deactivation request*.

GMT. Greenwich mean time.

Greenwich mean time (GMT). The mean solar time at the prime meridian (0 degrees longitude) of Greenwich, England. Greenwich mean time is sometimes called *Z time* or *Zulu time*.

Note: Although *Greenwich mean time (GMT)* and *coordinated universal time (UTC)* are sometimes used interchangeably, they are not synonyms. Greenwich mean time is an approximate time. Because the second is no longer defined in terms of astronomical phenomena, the preferred name for this time scale is *coordinated universal time (UTC)*.

group. In the NetView/PC program, to identify a set of application programs that are to run concurrently.

group ID (GID). (1) In RACF, a string of one to eight characters that identifies a group. The first character must be A through Z, #, $, or @. The rest can be A through Z, #, $, @, or 0 through 9. (2) In the AIX operating system, a number that corresponds to a specific group name. The group ID can often be substituted in commands that take a group name as a value.

H

handle. (1) In the Advanced DOS and OS/2 operating systems, a binary value created by the system that identifies a drive, directory, and file so that the file can be found and opened. (2) In the AIX operating system, a data structure that is a temporary local identifier for an object. Allocating a handle creates it. Binding a handle makes it identify an object at a specific location. (3) In OS/400 application programming interfaces, a variable that represents an object.

header. (1) System-defined control information that precedes user data. (2) The portion of a message that contains control information for the message such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

header file. Synonym for *include file*.

help panel. Information displayed by a system in response to a help request from a user.

hexadecimal. (1) Pertaining to a selection, choice, or condition that has 16 possible different values or states. (I) (2) Pertaining to a fixed-radix numeration system, with radix of 16. (I) (3) Pertaining to a system of numbers to the base 16; hexadecimal digits range from 0 through 9 and A through F, where A represents 10 and F represents 15.

high-level language (HLL). A programming language that does not reflect the structure of any particular computer or operating system.

High-Performance Routing (HPR). An addition to APPN that enhances data-routing performance and session reliability.

High-Performance Routing (HPR) node. An APPN end node or network node that includes High-Performance Routing support.

highlighting. Emphasizing a display element or segment by modifying its visual attributes. (I) (A)

hop. (1) In APPN, a portion of a route that has no intermediate nodes. It consists of only a single transmission group connecting adjacent nodes. (2) To the routing layer, the logical distance between two nodes in a network.

host. (1) In the Internet suite of protocols, an end system. The end system can be any workstation; it does not have to be a mainframe. (2) See *host processor*.

host ID. In the Internet suite of protocols, that part of the IP address that defines the host system on the network. The length of the host ID depends on the type of network or network class (A, B, or C).

host name. In the Internet suite of protocols, the name given to a machine. Sometimes, “host name” is used to mean *fully qualified domain name (FQDN)*; other times, it is used to mean the most specific subname of a fully qualified domain name. For example, if ralvm7.vnet.ibm.com is the fully qualified domain name, either of the following may be considered the host name:

ralvm7.vnet.ibm.com
ralvm7

host processor. (1) A processor that controls all or part of a user application network. (T) (2) In a network, the processing unit in which the data communication access method resides.

HPR. High-Performance Routing.

I

IBM Operating System/2 (OS/2). An IBM licensed program that can be used as the operating system for personal computers. The OS/2 licensed program can perform multiple tasks at the same time.

ICP. Internet Control Protocol.

ID. (1) Identifier. (2) Identification.

implicit focal point. An assigned focal point for which the nodes to be included in its sphere of control (SOC) are defined at the SOC nodes. The management

services capabilities exchange that brings a node into the sphere of control of an implicit focal point is initiated by the SOC node. Contrast with *explicit focal point*.

inactive. (1) Not operational. (2) Pertaining to a node or device not connected or not available for connection to another node or device. (3) In the AIX operating system, pertaining to a window that does not have an input focus. (4) In VTAM, the state of a resource or a major or minor node that has not been activated or for which the VARY INACT command has been issued. (5) Contrast with *active*. (6) See also *inoperative*.

inbound. In communications, data that is received from the network.

include file. A text file that contains declarations used by a group of functions, programs, or users. Synonymous with *header file*.

information (I) format. A format used for information transfer.

information (I) frame. A frame in I format used for numbered information transfer.

inhibited. In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is temporarily not ready to establish LU-LU sessions. An initiate request for a session with an inhibited LU will be rejected by the SSCP. The LU can separately indicate whether this applies to its ability to act as a primary logical unit (PLU) or a secondary logical unit (SLU). See also *disabled* and *enabled*.

initial program load (IPL). (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage at the beginning of a work day or after a system malfunction. (3) The process of loading system programs and preparing a system to run jobs. (4) Synonymous with *system restart* and *system startup*.

INITIATE. A network services request sent from a logical unit (LU) to a system services control point (SSCP) requesting that an LU-LU session be established.

inoperative. (1) The condition of a resource that has been active but is not currently active. A resource may be inoperative for reasons such as the following: (a) it may have failed, (b) it may have received an INOP request, or (c) it may be suspended while a reactivate command is being processed. (2) See also *inactive*.

installation. (1) In system development, preparing and placing a functional unit in position for use. (T) (2) A particular computing system, including the work it does and the people who manage it, operate it, apply it to problems, service it, and use the results it produces.

instance. In the AIX operating system, a concrete realization of an abstract object class. An instance of a widget or a gadget is a specific data structure that contains detailed appearance and behavioral information that is used to generate a specific graphical object on-screen at run time.

INT. Internal trace table.

interactive. (1) Pertaining to a program or system that alternately accepts input and then responds. An interactive system is conversational, that is, a continuous dialog exists between user and system. Contrast with *batch*. (2) Pertaining to the exchange of information between a user and a computer.

interface. (1) A shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics, as appropriate. The concept includes the specification of the connection of two devices having different functions. (T) (2) Hardware, software, or both, that links systems, programs, or devices.

intermediate session routing (ISR). A type of routing function within an APPN network node that provides session-level flow control and outage reporting for all sessions that pass through the node but whose end points are elsewhere.

Internet Control Protocol (ICP). The VIrtual NEtworking System (VINES) protocol that provides exception notifications, metric notifications, and PING support. See also *RouTing update Protocol (RTP)*.

Internet Protocol (IP). A connectionless protocol that routes data through a network or interconnected networks. IP acts as an intermediary between the higher protocol layers and the physical network. However, this protocol does not provide error recovery and flow control and does not guarantee the reliability of the physical network.

IP. Internet Protocol.

IP filter. In the Internet suite of protocols, a set of rules based on IP addressing that control whether one host can access another host through a firewall.

IPL. (1) Initial program loader. (A) (2) Initial program load.

ISR. Intermediate session routing.

item. In CCP, any of the components, such as communication controllers, lines, cluster controllers, and terminals, that comprise an IBM 3710 Network Controller configuration.

K

keylock. In Communications Manager/2, a function that controls access to configuration data and some advanced functions. See also *lock*, *master key*, and *service key*.

keyword. (1) In programming languages, a lexical unit that, in certain contexts, characterizes some language construct; for example, in some contexts, IF characterizes an if-statement. A keyword normally has the form of an identifier. (I) (2) One of the predefined words of an artificial language. (A) (3) A significant and informative word in a title or document that describes the content of that document. (4) A name or symbol that identifies a parameter. (5) The part of a command operand that consists of a specific character string (such as DSNAME=). (6) See also *keyword operand*. (7) Contrast with *positional operand*.

keyword operand. (1) An operand that consists of a keyword followed by one or more values (such as DSNAME=HELLO). (2) Contrast with *positional operand*. (3) See also *definition statement*.

keyword parameter. A parameter that consists of a keyword followed by one or more values.

L

LAN. Local area network.

latency. The time interval between the instant at which an instruction control unit initiates a call for data and the instant at which the actual transfer of the data starts. (T)

LDT. Local descriptor table.

LEN. Low-entry networking.

LEN end node. See *low-entry networking (LEN) end node*.

LEN node. A node that supports independent LU protocols but does not support CP-CP sessions. It may be a peripheral node attached to a boundary node in a subarea network, an end node attached to an APPN network node in an APPN network, or a peer-connected node directly attached to another LEN node or APPN end node. See also *low-entry networking (LEN) end node*.

LFSID. Local-form session identifier.

limited resource. A connection facility that causes a session traversing it to be terminated if no session activity is detected for a specified period of time. See also *limited-resource session*.

limited-resource link. A link defined by the node operator to be a limited resource, that is, a resource to remain active only when being used. Limited-resource links are deactivated if no session activity has been detected for a specified period of time. See also *limited-resource session*.

limited-resource session. A session that traverses a limited-resource link. This session is terminated if no session activity is detected for a specified period of time.

line. (1) The portion of a data circuit external to data circuit-terminating equipment (DCE), that connects the DCE to a data switching exchange (DSE), that connects a DCE to one or more other DCEs, or that connects a DSE to another DSE. (I) (2) Synonymous with *channel* and *circuit*.

line control discipline. Synonym for *link protocol* and *protocol*.

line discipline. Synonym for *link protocol* and *protocol*.

line speed. The number of binary digits that can be sent over a telecommunication line in one second, expressed in bits per second (bps).

link. (1) The combination of the link connection (the transmission medium) and two link stations, one at each end of the link connection. A link connection can be shared among multiple links in a multipoint or token-ring configuration. (2) To interconnect items of data or portions of one or more computer programs: for example, the linking of object programs by a linkage editor, linking of data items by pointers. (T) (3) In SNA, synonymous with *data link*.

link-attached. (1) Pertaining to devices that are connected to a controlling unit by a data link. (2) Contrast with *channel-attached*. (3) Synonymous with *remote*.

link connection. (1) The physical equipment providing two-way communication between one link station and one or more other link stations; for example, a telecommunication line and data circuit-terminating equipment (DCE). (2) In SNA, synonymous with *data circuit*.

link connection network. Synonym for *connection network*.

link connection segment. A portion of the configuration that is located between two resources listed consecutively in the service point command service (SPCS) query link configuration request list.

link level. (1) A part of Recommendation X.25 that defines the link protocol used to get data into and out of the network across the full-duplex link connecting the subscriber's machine to the network node. LAP and LAPB are the link access protocols recommended by the CCITT. (2) See *data link level*.

link protocol. (1) The rules for sending and receiving data at the link level. (2) Synonymous with *line control discipline* and *line discipline*.

link station. (1) The hardware and software components within a node representing a connection to an adjacent node over a specific link. For example, if node A is the primary end of a multipoint line that connects to three adjacent nodes, node A will have three link stations representing the connections to the adjacent nodes. (2) In VTAM, a named resource within an APPN or a subarea node that represents the connection to another APPN or subarea node that is attached by an APPN or a subarea link. In the resource hierarchy in a subarea network, the link station is subordinate to the subarea link. (3) See also *adjacent link station (ALS)*.

link station role. In SNA, the role that a local node assumes for a given link. Possible roles are primary (or controlling), secondary, or negotiable.

link status (LS). Information maintained by local and remote modems.

load. (1) To bring all or part of a computer program into memory from auxiliary storage so that the computer can run the program. (2) To place a diskette into a diskette drive.

local. (1) Pertaining to a device accessed directly without use of a telecommunication line. (2) Contrast with *remote*. (3) Synonym for *channel-attached*.

local area network (LAN). (1) A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. (T) (2) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network. (3) See also *Ethernet* and *token ring*. (4) Contrast with *metropolitan area network (MAN)* and *wide area network (WAN)*.

local descriptor table (LDT). In the OS/2 operating system, a table that contains access information about the code and data segments for which a process has addressability.

local directory database. That set of resources (LUs) in the network known at a particular node. The resources included are all those in the node's domain as well as any cache entries.

local-form session identifier (LFSID). A dynamically assigned value used at a type 2.1 node to identify traffic for a particular session using a given transmission group (TG). The LFSID is encoded in the ODAI, OAF′, and DAF′ fields of the transmission headers that accompany session messages exchanged over the TG.

local topology database. A database in an APPN or LEN node containing an entry for each transmission group (TG) having at least one end node for an endpoint. In an end node, the database has one entry for each TG connecting to the node. In a network node, the database has an entry for each TG connecting the network node to an end node. Each entry describes the current characteristics of the TG that it represents. A network node has both a local and a network topology database while an end node has only a local topology database.

Locate. Synonym for *Locate/CD-Initiate*.

Locate search message. Synonym for *Locate/CD-Initiate.*

Locate/CD-Initiate. (1) An abbreviated term for a message exchanged between APPN nodes that contains one of the following sets of general data stream (GDS) variables:

A Locate, a Find Resource, and a Cross-Domain Initiate GDS variable used for a network search request

A Locate, a Found Resource, and a Cross-Domain Initiate GDS variable used for a search reply when a network resource has been located

These message structures correspond to the CP components that perform the search of the distributed network directory and establish the session. The Locate GDS variable contains information used to control the delivery of the search messages in the network. The Find and Found GDS variables contain information used in the directories: origin cache data (control point information) and search arguments (destination LU name), and located resource information, respectively. The Cross-Domain Initiate GDS variable contains endpoint TG vector information to be used in selecting the route for the session. The length of the Locate/CD-Initiate message is limited to 1024 bytes. (2) Synonymous with *Locate* and *Locate search message*.

lock. (1) The means by which integrity of data is ensured by preventing more than one user from accessing or changing the same data or object at the same time. (2) In Communications Manager/2, a password-protection system that can be used to prevent access to some advanced functions. See also *keylock*.

logged-on operator. (1) A NetView operator station task that requires a terminal and a logged-on user. (2) Contrast with *autotask*.

logical unit (LU). A type of network accessible unit that enables users to gain access to network resources and communicate with each other.

low-entry networking (LEN). A capability of nodes to attach directly to one another using basic peer-to-peer protocols to support multiple and parallel sessions between logical units.

low-entry networking (LEN) end node. A LEN node receiving network services from an adjacent APPN network node.

low-entry networking (LEN) node. A node that provides a range of end-user services, attaches directly to other nodes using peer protocols, and derives network services implicitly from an adjacent APPN network node, that is, without the direct use of CP-CP sessions.

LS. Link status.

LU. Logical unit.

LU-LU session. A logical connection between two logical units (LUs) in an SNA network that typically provides communication between two users.

LU type. The classification of an LU in terms of the specific subset of SNA protocols and options it supports for a given session, namely:

The mandatory and optional values allowed in the session activation request

The usage of data stream controls, function management headers (FMHs), request unit parameters, and sense data values

Presentation services protocols such as those associated with FMH usage

LU types 0, 1, 2, 3, 4, 6.1, 6.2, and 7 are defined.

LU 6.2. A type of logical unit that supports general communication between programs in a distributed processing environment. LU 6.2 is characterized by (a) a peer relationship between session partners, (b) efficient utilization of a session for multiple transactions, (c) comprehensive end-to-end error processing, and (d) a generic application programming interface (API) consisting of structured verbs that are mapped into a product implementation.

LU 6.2 verb. A syntactical unit in the LU 6.2 application programming interface representing an operation.

LUS. Logical unit services.

M

MAC. Medium access control.

macroinstruction. (1) An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language and that may also specify values for parameters in the replaced instructions. (T) (2) In assembler programming, an assembler language statement that causes the assembler to process a predefined set of statements called a macro definition. The statements normally produced from the macro definition replace the macroinstruction in the program. (3) See also *definition statement*.

maintenance analysis procedure (MAP). A maintenance document that gives an IBM service representative a step-by-step procedure for tracing a symptom to the cause of a failure.

major node. In VTAM, a set of resources that can be activated and deactivated as a group. See also *minor node*.

MAN. Metropolitan area network.

Management Information Base (MIB). (1) A collection of objects that can be accessed by means of a network management protocol. (2) A definition for management information that specifies the information available from a host or gateway and the operations allowed. (3) In OSI, the conceptual repository of management information within an open system.

management services (MS). (1) One of the types of network services in control points (CPs) and physical units (PUs). Management services are the services provided to assist in the management of SNA networks, such as problem management, performance and accounting management, configuration management, and change management. (2) Services that assist in the management of systems and networks in areas such as problem management, performance management, business management, operations management, configuration management, and change management.

management services focal point (MSFP). For any given management services discipline (for example, problem determination or response time monitoring), the control point that is responsible for that type of network management data for a sphere of control. This responsibility may include collecting, storing, or displaying the data, or all of these. (For example, a problem determination focal point is a control point that collects, and that may store or display, problem determination data.)

management services unit (MSU). A generic term for major-vector encoded management services data, regardless of the method used to transport the data. Thus, the management services unit includes major vectors transported within the network management vector transport (NMVT), the control point management services unit (CP-MSU), or the multiple-domain support message unit (MDS-MU).

manager. (1) In systems management, a user that, for a particular interaction, has assumed a manager role. (2) An entity that monitors or controls one or more managed objects by (a) receiving notifications regarding the objects and (b) requesting management operations to modify or query the objects. (3) A system that assumes a manager role.

map. In NetView for AIX, a database represented by a set of related submaps that provide a graphical and hierarchical presentation of a network and its systems.

MAP. Maintenance analysis procedure.

mapped conversation. An LU 6.2 conversation type specified by the allocating transaction program. Transaction programs using a mapped conversation can exchange messages of arbitrary format regardless of the underlying data stream. System-defined or user-defined mappers can perform data transformation for the transaction programs. See also *conversation*. Contrast with *basic conversation*.

mapping. The process of converting data that is transmitted in one format by the sender into the data format that can be accepted by the receiver.

master key. In Communications Manager/2, the access password used to unlock the keylock for Communications Manager/2. A user who has access to the master key can also change the service key.

Maximize. A choice that enlarges a window to its largest possible size.

MDS. Multiple-domain support.

MDS-MU. Multiple-domain support message unit.

medium. (1) A physical carrier of electrical energy. (2) A physical material in or on which data may be represented.

medium access control (MAC). In LANs, the sublayer of the data link control layer that supports medium-dependent functions and uses the services of the physical layer to provide services to the logical link control (LLC) sublayer. The MAC sublayer includes the method of determining when a device has access to the transmission medium.

medium access control (MAC) protocol. In a local area network, the protocol that governs access to the transmission medium, taking into account the topological aspects of the network, in order to enable the exchange of data between data stations. (T)

medium access control (MAC) sublayer. In a local area network, the part of the data link layer that applies a medium access method. The MAC sublayer supports topology-dependent functions and uses the services of the physical layer to provide services to the logical link control sublayer. (T)

memory. All of the addressable storage space in a processing unit and other internal storages that is used to execute instructions. (T)

menu. (1) A list of options displayed to the user by a data processing system, from which the user can select an action to be initiated. (T) (2) In text processing, a list of choices displayed to the user by a text processor from which the user can select an action to be initiated. (T) (3) A list of choices that can be applied to an object. A menu can contain choices that are not available for selection in certain contexts. Those choices are indicated by reduced contrast.

message. (1) An assembly of characters and sometimes control codes that is transferred as an entity from an originator to one or more recipients. A message consists of two parts: envelope and content. (T) (2) A communication sent from a person or program to another person or program. (3) In VTAM, the amount of function management data (FMD) transferred to VTAM by the application program with one SEND request.

message unit (MU). In SNA, the unit of data processed by any layer; for example, a basic information unit (BIU), a path information unit (PIU), or a request/response unit (RU).

method. In the NetView program, the code that runs within the Resource Object Data Manager (RODM) address space. Methods are used to implement behavior specified by an operation.

metric. In Internet communications, a value, associated with a route, which is used to discriminate between multiple exit or entry points to the same autonomous system. The route with the lowest metric is preferred.

metropolitan area network (MAN). A network formed by the interconnection of two or more networks which may operate at higher speed than those networks, may cross administrative boundaries, and may use multiple access methods. (T) Contrast with *local area network (LAN)* and *wide area network (WAN)*.

MIB. (1) MIB module. (2) Management Information Base.

migration. The installation of a new version or release of a program to replace an earlier version or release.

minor node. In VTAM, a uniquely defined resource within a major node.

mixed-media multilink transmission group (MMMLTG). See *transmission group (TG)*.

MLTG. Multilink transmission group.

MMMLTG. Mixed-media multilink transmission group.

mode. See *mode name*.

mode name. The name used by the initiator of a session to designate the characteristics desired for the session, such as traffic pacing values, message-length limits, sync point and cryptography options, and the class of service within the transport network.

modem (modulator/demodulator). (1) A functional unit that modulates and demodulates signals. One of the functions of a modem is to enable digital data to be transmitted over analog transmission facilities. (T) (A) (2) A device that converts digital data from a computer to an analog signal that can be transmitted on a telecommunication line, and converts the analog signal received to data for the computer.

module. A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to or output from an assembler, compiler, linkage editor, or executive routine. (A)

MS. Management services.

MSFP. Management services focal point.

MSG. Console messages.

MSU. Management services unit.

MU. Message unit.

multicast. (1) Transmission of the same data to a selected group of destinations. (T) (2) A special form of broadcast in which copies of a packet are delivered to only a subset of all possible destinations. (3) Contrast with *broadcast*.

multilink transmission group (MLTG). See *transmission group (TG)*.

multiple-domain support (MDS). A technique for transporting management services data between management services function sets over LU-LU and CP-CP sessions. See also *multiple-domain support message unit (MDS-MU)*.

multiple-domain support message unit (MDS-MU). The message unit that contains management services data and flows between management services function sets over the LU-LU and CP-CP sessions used by multiple-domain support. This message unit, as well as the actual management services data that it contains, is in general data stream (GDS) format. See also *control point management services unit (CP-MSU)*, *management services unit (MSU)*, and *network management vector transport (NMVT)*.

N

NAU. (1) Network accessible unit. (2) Network addressable unit.

NC. Network control.

negative response (NR). In SNA, a response indicating that a request did not arrive successfully or was not processed successfully by the receiver. Contrast with *positive response*.

negotiation. The process of deciding what packet size to transmit between a network and a 3710 Network Controller.

NETID. See *network identifier*.

NetView-NetView task (NNT). The task under which a cross-domain NetView operator session runs. See *operator station task*.

network. (1) An arrangement of nodes and connecting branches. (T) (2) A configuration of data processing devices and software connected for information interchange. (3) A group of nodes and the links interconnecting them.

network accessible unit (NAU). A logical unit (LU), physical unit (PU), control point (CP), or system services control point (SSCP). It is the origin or the destination of information transmitted by the path control network. Synonymous with *network addressable unit*.

network address. (1) According to ISO 7498-3, a name, unambiguous within the OSI environment, that identifies a set of network service access points. (2) An identifier for a node, station, or unit of equipment in a network. (3) In a subarea network, an address, consisting of subarea and element fields, that identifies a link, link station, physical unit, logical unit, or system services control point. Subarea nodes use network addresses; peripheral nodes use local addresses or local-form session identifiers (LFSIDs). The boundary function in the subarea node to which a peripheral node is attached transforms local addresses or LFSIDs to

network addresses and vice versa. Contrast with *network name*.

network addressable unit (NAU). Synonym for *network accessible unit*.

network administrator. A person who manages the use and maintenance of a network.

network architecture. The logical structure and operating principles of a computer network. (T)

Note: The operating principles of a network include those of services, functions, and protocols.

network congestion. An undesirable overload condition caused by traffic in excess of what a network can handle.

network control (NC). In SNA, a request/response unit (RU) category used for requests and responses exchanged between physical units (PUs) for such purposes as activating and deactivating explicit and virtual routes and sending load modules to adjust peripheral nodes. See also *data flow control*, *function management data*, and *session control*.

network control program. A program, generated by the user from a library of IBM-supplied modules, that controls the operation of a communication controller.

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

network directory database. Synonym for *distributed directory database*.

network identifier. (1) In TCP/IP, that part of the IP address that defines a network. The length of the network ID depends on the type of network class (A, B, or C). (2) A 1- to 8-byte customer-selected name or an 8-byte IBM-registered name that uniquely identifies a specific subnetwork. (3) In MPTN architecture, the address qualifier of a transport provider address that identifies a group of nodes according to the network in which they reside.

network management. The process of planning, organizing, and controlling a communication-oriented data processing or information system.

network management vector transport (NMVT). A management services request/response unit (RU) that flows over an active session between physical unit management services and control point management services (SSCP-PU session).

network name. (1) The symbolic identifier by which users refer to a network accessible unit, a link, or a link station within a given subnetwork. In APPN networks, network names are also used for routing purposes. Contrast with *network address*. (2) In a multiple-domain network, the name of the APPL statement defining a VTAM application program. The network name must be unique across domains. Contrast with *ACB name*. See *uninterpreted name*.

network node (NN). See *Advanced Peer-to-Peer Networking (APPN) network node*.

network node domain. An APPN network-node control point, its attached links, the network resources for which it answers directory search requests (namely, its local LUs and adjacent LEN end nodes), the adjacent APPN end nodes with which it exchanges directory search requests and replies, and other resources (such as a local storage device) associated with its own node or an adjacent end node for which it provides management services.

network node server. An APPN network node that provides network services for its local LUs and client end nodes.

network operator. (1) A person who controls the operation of all or part of a network. (2) In a multiple-domain network, a person or program responsible for controlling all domains. (3) See also *domain operator*.

network-qualified name. In SNA, a name that uniquely identifies a specific resource (such as an LU or a CP) within a specific network. It consists of a network identifier and a resource name, each of which is a 1- to 8-byte symbol string. Synonymous with *fully qualified name*.

network topology database. The representation of the current connectivity between the network nodes within an APPN network. It includes (a) entries for all network nodes and the transmission groups interconnecting them and (b) entries for all virtual routing nodes to which network nodes are attached.

NMVT. Network management vector transport.

NN. Network node.

NNCP. Network node control point.

no response. In SNA, a protocol requested in the form-of-response-requested field of the request header that directs the receiver of the request not to return any response, regardless of whether or not the request is received and processed successfully. Contrast with *definite response* and *exception response*.

node. (1) In a network, a point at which one or more functional units connect channels or data circuits. (I) (2) In network topology, the point at an end of a

branch. (T) (3) The representation of a state or an event by means of a point on a diagram. (A) (4) In a tree structure, a point at which subordinate items of data originate. (A) (5) Any device, attached to a network, that transmits and receives data. (6) An endpoint of a link or a junction common to two or more links in a network. Nodes can be processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities. (7) In VTAM, a point in a network defined by a symbolic name. See *major node* and *minor node*.

node name. In VTAM, the symbolic name assigned to a specific major or minor node during network definition.

node type. A designation of a node according to the protocols it supports or the role it plays in a network. Node type was originally denoted numerically (as 1, 2.0, 2.1, 4, and 5) but is now characterized more specifically by protocol type (APPN network node, LEN node, subarea node, and interchange node, for example) because type 2.1 nodes and type 5 nodes support multiple protocol types and roles.

normal flow. In SNA, a data flow designated in the transmission header (TH) that is used primarily to carry end-user data. The rate at which requests flow on the normal flow can be regulated by session-level pacing. Normal and expedited flows move in both the primary-to-secondary and secondary-to-primary directions. Contrast with *expedited flow*.

notification. (1) An unscheduled, spontaneously generated report of an event that has occurred. (2) In systems management, information emitted by a managed object relating to an event that has occurred within the managed object, such as a threshold violation or a change in configuration status.

NOTIFY. A network services request that is sent by a system services control point (SSCP) to a logical unit (LU) to inform the LU of the status of a procedure requested by the LU.

O

ODAI. Origin-Destination Assignor indicator, a bit in a FID2 transmission header used to divide the address space so that an address space manager (ASM) in one node may use all possible combinations of OAF′, DAF′ with the ODAI having one setting and the ASM in the adjacent node may use all possible combinations of OAF′, DAF′ with the ODAI having the complementary setting.

OEM. Original equipment manufacturer.

Off. A choice that appears in the cascaded menu from the Refresh choice. It sets the refresh function to off.

offset. The number of measuring units from an arbitrary starting point in a record, area, or control block, to some other point.

OIA. Operator information area.

OK. A push button that accepts the information in a window and closes it. If the window contains changed information, those changes are applied before the window is closed.

On. A choice that appears in a cascaded menu from the Refresh choice. It immediately refreshes the view in a window.

open. (1) A break in an electrical circuit. (2) To make an adapter ready for use.

Open. A choice that leads to a window in which users can select the object they want to open.

operable time. The time during which a functional unit would yield correct results if it were operated. (I) (A) Synonymous with *uptime*.

operand. (1) An entity on which an operation is performed. (I) (2) That which is operated upon. An operand is usually identified by an address part of an instruction. (A) (3) Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor. (4) An expression to whose value an operator is applied. See also *definition statement*, *keyword*, *keyword parameter*, and *parameter*.

operating system (OS). Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible. (T)

operating time. That part of operable time during which a functional unit is operated. (A)

operation. In object-oriented design or programming, a service that can be requested at the boundary of an object. Operations include modifying an object or disclosing information about an object.

operator. (1) In a language statement, the lexical entity that indicates the action to be performed on operands. See also *definition statement*. (2) A person or program responsible for managing activities controlled by a given piece of software such as MVS, the NetView program, or IMS. (3) A person who operates a device. (4) A person who keeps a system running. (5) See also *autotask*, *logged-on operator*, *network operator*, and *operator station task*.

operator information area (OIA). The area near the bottom of the display area where terminal or system status information is displayed.

operator station task (OST). The NetView task that establishes and maintains the online session with the network operator. There is one operator station task for each network operator who logs on to the NetView program. See *NetView-NetView task*.

origin. An external logical unit (LU) or application program from which a message or other data originates. See also *destination*.

original equipment manufacturer (OEM). A manufacturer of equipment that may be marketed by another manufacturer.

OS. Operating system.

outbound. In communications, data that is transmitted to the network.

overlay. A collection of predefined data, such as lines, shading, text, boxes, or logos, that can be merged with variable data on a page while printing.

P

PAC. Privilege Attribute Certificate.

pacing. (1) A technique by which a receiving component controls the rate of transmission of a sending component to prevent overrun or congestion. (2) See *receive pacing*, *send pacing*, *session-level pacing*, and *virtual route (VR) pacing*. (3) See also *flow control*.

pacing group. Synonym for *pacing window*.

pacing response. In SNA, an indicator that signifies the readiness of a receiving component to accept another pacing group. The indicator is carried in a response header (RH) for session-level pacing and in a transmission header (TH) for virtual route pacing.

pacing window. (1) The path information units (PIUs) that can be transmitted on a virtual route before a virtual-route pacing response is received, indicating that the virtual route receiver is ready to receive more PIUs on the route. (2) The requests that can be transmitted on the normal flow in one direction on a session before a session-level pacing response is received, indicating that the receiver is ready to accept the next group of requests. (3) Synonymous with *pacing group*.

packet assembler/disassembler (PAD). A functional unit that enables data terminal equipment (DTEs) not equipped for packet switching to access a packet switched network. (T) (A)

packet internet groper (PING). (1) In Internet communications, a program used in TCP/IP networks to test the ability to reach destinations by sending the destinations an Internet Control Message Protocol (ICMP) echo request and waiting for a reply. (2) In communications, a test of reachability.

packet level. (1) The packet format and control procedures for exchange of packets containing control information and user data between data terminal equipment (DTE) and data circuit-terminating equipment (DCE). See also *data link level* and *physical level*. (2) A part of Recommendation X.25 that defines the protocol for establishing logical connections between two DTEs and for transferring data on these connections.

PAD. Packet assembler/disassembler.

page. (1) In a virtual storage system, a fixed-length block that has a virtual address and is transferred as a unit between real storage and auxiliary storage. (I) (A) (2) A printed form. (3) The information displayed at the same time on the screen of a display device. (4) In VSE, a fixed-length block of instructions, data, or both that can be located in processor storage or in the page data set on disk. (5) To replace the information displayed on the screen with prior or subsequent information from the same file.

panel. (1) See *window*. (2) A formatted display of information that appears on a display screen. See *help panel* and *task panel*. (3) In computer graphics, a display image that defines the locations and characteristics of display fields on a display surface.

parallel. (1) Pertaining to a process in which all events occur within the same interval of time, each handled by a separate but similar functional unit; for example, the parallel transmission of the bits of a computer word along the lines of an internal bus. (T) (2) Pertaining to concurrent or simultaneous operation of two or more devices or to concurrent performance of two or more activities in a single device. (A) (3) Pertaining to concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels. (A) (4) Pertaining to the simultaneity of two or more processes. (A) (5) Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts. (A) (6) Contrast with *serial*.

parallel sessions. Two or more concurrently active sessions between the same two network accessible units (NAUs) using different pairs of network addresses or local-form session identifiers. Each session can have independent session parameters.

parallel transmission groups. Multiple transmission groups between adjacent nodes, with each group having a distinct transmission group number.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (I) (A) (2) In Basic CUA architecture, a variable used in conjunction with a command to affect its result. (3) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (4) Data passed to a program or procedure by a user or another program, namely as an operand in a language statement, as an item in a menu, or as a shared data structure. (5) See also *keyword*, *keyword parameter*, and *operand*.

parameters. In NetDA/2, the set of restrictions that affect only the output of a network design. A change in a parameter value does not change the input to the network design. Contrast with *constraints*.

parent. A process that has spawned a child process using the fork primitive. Contrast with *child*.

parse. To analyze the operands entered with a command and create a parameter list for the command processor from the information.

partner-LU verification. For logical unit (LU) 6.2, a three-flow exchange between two LUs with each LU using an LU-LU password and the Data Encryption Standard (DES) algorithm. The three-flow exchange is the LU-LU verification. See *end-user verification*.

password. (1) A value used in authentication or a value used to establish membership in a set of people having specific privileges. (2) A unique string of characters known to a computer system and to a user, who must specify the character string to gain access to a system and to the information stored within it. (3) In computer security, a string of characters known to the computer system and a user, who must specify it to gain full or limited access to a system and to the data stored within it.

path. (1) In a network, any route between any two nodes. A path may include more than one branch. (T) (2) The route used to locate files; the storage location of a file. A fully qualified path lists the drive identifier, directory name, subdirectory name (if any), and file name with the associated extension. (3) The series of transport network components (path control and data link control) that are traversed by the information exchanged between two network accessible units. See also *explicit route (ER)*, *route extension (REX)*, and *virtual route (VR)*. (4) In VTAM when defining a switched major node, a potential dial-out port that can be used to reach that node. (5) In the NetView/PC program, a complete line in a configuration that contains all of the resources in the service point command service (SPCS) query link configuration request list.

path control (PC). The function that routes message units between network accessible units in the network and provides the paths between them. It converts the basic information units (BIUs) from transmission control (possibly segmenting them) into path information units (PIUs) and exchanges basic transmission units containing one or more PIUs with data link control. Path control differs by node type: some nodes (APPN nodes, for example) use locally generated session identifiers for routing, and others (subarea nodes) use network addresses for routing.

path information unit (PIU). A message unit consisting of a transmission header (TH) alone, or a TH followed by a basic information unit (BIU) or a BIU segment.

PC. (1) Personal computer. (2) Path control.

PCID. Procedure-correlation identifier.

peer. In network architecture, any functional unit that is in the same layer as another entity. (T)

pending active session. In VTAM, the state of an LU-LU session recorded by the system services control point (SSCP) when it finds both logical units (LUs) available and has sent a CINIT request to the primary logical unit (PLU) of the requested session.

peripheral PU. In SNA, a physical unit in a peripheral node. Contrast with *subarea PU*.

personal computer (PC). (1) A microcomputer primarily intended for stand-alone use by an individual. (T) (2) A desktop, floor-standing, or portable microcomputer that usually consists of a system unit, a display monitor, a keyboard, one or more diskette drives, internal fixed-disk storage, and an optional printer. PCs are designed primarily for stand-alone operation but may be connected to mainframes or networks.

physical circuit. A circuit established without multiplexing. See also *data circuit*. Contrast with *virtual circuit*.

physical connection. (1) A connection that establishes an electrical circuit. (2) A point-to-point or multipoint connection. (3) Synonymous with *connection*.

physical level. In X.25, the mechanical, electrical, functional, and procedural media used to activate, maintain, and deactivate the physical link between the data terminal equipment (DTE) and the data

circuit-terminating equipment (DCE). See *data link level* and *packet level*.

physical unit (PU). (1) The component that manages and monitors the resources (such as attached links and adjacent link stations) associated with a node, as requested by an SSCP via an SSCP-PU session. An SSCP activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links. This term applies to type 2.0, type 4, and type 5 nodes only. (2) See also *peripheral PU* and *subarea PU*.

physical unit (PU) services. In SNA, the components within a physical unit (PU) that provide configuration services and maintenance services for SSCP-PU sessions.

PING. Packet internet groper.

PIP. Program initialization parameters.

pipe. To direct data so that the output from one process becomes the input to another process.

PLU. Primary logical unit.

point-to-point. Pertaining to data transmission between two locations without the use of any intermediate display station or computer.

pointer. (1) A data element that indicates the location of another data element. (T) (2) An identifier that indicates the location of an item of data. (A)

polling. (1) On a multipoint connection or a point-to-point connection, the process whereby data stations are invited, one at a time, to transmit. (I) (2) Interrogation of devices for such purposes as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (A)

pop. To remove an item from the top of a pushdown list. Contrast with *push*.

POP. Post Office Protocol.

port. (1) An access point for data entry or exit. (2) A connector on a device to which cables for other devices such as display stations and printers are attached. (3) The representation of a physical connection to the link hardware. A port is sometimes referred to as an adapter; however, there can be more than one port on an adapter. There may be one or more ports controlled by a single DLC process. (4) In the Internet suite of protocols, a 16-bit number used to communicate between TCP or the User Datagram Protocol (UDP) and a higher-level protocol or application. Some protocols, such as File Transfer Protocol (FTP) and Simple Mail Transfer Protocol (SMTP), use the same well-known port number in all TCP/IP implementations. (5) An abstraction used by transport protocols to distinguish among multiple destinations within a host machine. (6) Synonymous with *socket*.

port number. In Internet communications, the identification of an application entity to the transport service.

positional operand. (1) An operand in a language statement that has a fixed position. (2) Contrast with *keyword operand*. (3) See also *definition statement*.

positive response. In SNA, a response indicating that a request was received and processed. Contrast with *negative response*.

Post Office Protocol (POP). A protocol used for exchanging network mail and accessing mailboxes.

Prepare. A presentation services header that flows as part of commit processing, indicating the partner has begun the first phase of the two-phase commit process.

presentation space ID (PSID). In Communications Manager/2, synonym for *short name*.

primary focal point. A focal point understood to be the preferred source of management services support for a particular category. Contrast with *backup focal point*.

primary logical unit (PLU). In SNA, the logical unit (LU) that sends the BIND to activate a session with its partner LU. Contrast with *secondary logical unit (SLU)*.

Privilege Attribute Certificate (PAC). In a Distributed Computing Environment (DCE), a certified set of access privileges that can be presented by a user or an administrator to establish access rights to objects.

PRM. Protected-resource manager.

problem determination. The process of determining the source of a problem; for example, a program component, machine failure, telecommunication facilities, user or contractor-installed programs or equipment, environmental failure such as a power loss, or user error.

procedure. (1) In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a procedure call. (I) (2) The description of the course of action taken for the solution of a problem. (A)

procedure-correlation identifier (PCID). In SNA, a value used to correlate all requests and replies associated with a given procedure.

process. (1) To perform operations on data in a process. (I) (A) (2) In data processing, the course

of events that occurs during the execution of all or part of a program. (T) (3) A course of the events defined by its purpose or by its effect, achieved under given conditions. (4) Any operation or combination of operations on data. (5) A function being performed or waiting to be performed.

process identification number (process ID). A unique number assigned to a process by the operating system. The number is used internally by processes to communicate.

processor. In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit. (T)

product-set identification (PSID). (1) In SNA, a technique for identifying the hardware and software products that implement a network component. (2) A management services common subvector that transports the information described in definition (1).

profile. Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

program. (1) A sequence of instructions suitable for processing by a computer. Processing may include the use of an assembler, a compiler, an interpreter, or a translator to prepare the program for execution, as well as to execute it. (I) (2) In programming languages, a logical assembly of one or more interrelated modules. (I) (3) To design, write, and test computer programs. (I) (A)

program initialization parameters (PIP). The initial parameter values passed to a target program as input or used to set up the process environment.

program temporary fix (PTF). A temporary solution or bypass of a problem diagnosed by IBM in a current unaltered release of the program.

programmable operator facility (PROP). A VM facility that allows remote control of a virtual machine by intercepting messages directed for that machine and taking preprogrammed action.

PROP. Programmable operator facility.

protected resource. A resource, for example, a database, that can be modified only in accordance with sync point protocols.

protected-resource manager (PRM). See *protection manager*.

protection manager. A system component, associated with one or more protected resources, that interacts with sync point services to carry out two-phase commit protocols. A protection manager implements the parts of the protocol that are specific to a particular resource type. For example, the protection manager associated with a protected conversation builds and sends PS headers in response to instructions from sync point services. Two types of protection managers are defined: conversational resource protection managers and local (nonconversational) resource protection managers. Together, these resource protection managers are referred to as protected-resource managers.

protocol. (1) A set of semantic and syntactic rules that determine the behavior of functional units in achieving communication. (I) (2) In Open Systems Interconnection architecture, a set of semantic and syntactic rules that determine the behavior of entities in the same layer in performing communication functions. (T) (3) In SNA, the meanings of, and the sequencing rules for, requests and responses used for managing the network, transferring data, and synchronizing the states of network components. Synonymous with *line control discipline* and *line discipline*. See *bracket protocol* and *link protocol*.

PSID. (1) Product-set identification. (2) Presentation space ID.

PTF. Program temporary fix.

PU. Physical unit.

PU type. (1) Deprecated term for *node type*. (2) The type of physical unit in a node.

push. To add an item to the top of a pushdown list. Contrast with *pop*.

pushdown list. (1) A list constructed and maintained so that the next data element to be retrieved is the most recently stored. (T) (2) Synonymous with *stack*.

PUT. Program update tape.

Q

queue. (1) A list constructed and maintained so that the next data element to be retrieved is the one stored first. (T) (2) A line or list of items waiting to be processed; for example, work to be performed or messages to be displayed. (3) To arrange in or form a queue.

quiesce. (1) To end a process by allowing operations to complete normally. (2) To request that a node stop sending synchronous-flow messages.

R

Rapid Transport Protocol (RTP). The processing used by the endpoints of an RTP connection for route setup, nondisruptive route switching, adaptive-route-based congestion control, and error recovery.

Rapid Transport Protocol (RTP) connection. In High-Performance Routing (HPR), the connection established between the endpoints of the route to transport session traffic.

RAR. Route addition resistance.

read-only memory (ROM). Memory in which stored data cannot be modified by the user except under special conditions.

reassembly. In communications, the process of putting segmented packets back together after they have been received.

receive pacing. In SNA, the pacing of message units that a component is receiving. Contrast with *send pacing*.

record. (1) In programming languages, an aggregate that consists of data objects, possibly with different attributes, that usually have identifiers attached to them. In some programming languages, records are called structures. (I) (2) A set of data treated as a unit. (T) (3) A set of one or more related data items grouped for processing. (4) In VTAM, the unit of data transmission for record mode. A record represents whatever amount of data the transmitting node chooses to send.

recording filter. In the NetView program, the function that determines which events, statistics, and alerts are stored on a database.

release. (1) A distribution of a new product or new function and APAR fixes for an existing product. Normally, programming support for the prior release is discontinued after some specified period of time following availability of a new release. The first version of a product is announced as Release 1, Modification Level 0. (2) In VTAM, to relinquish control of resources (communication controllers or physical units). See also *resource takeover*. Contrast with *acquire*.

remote. (1) Pertaining to a system, program, or device that is accessed through a telecommunication line. (2) Synonym for *link-attached*. (3) Contrast with *local*.

remote host. Any host on a network except the host at which a particular operator is working. Synonymous with *foreign host*.

remove. In the IBM Token-Ring Network, to take an attaching device off the ring.

request header (RH). The control information that precedes a request unit (RU). See also *request/response header (RH)*.

request unit (RU). A message unit that contains control information, end-user data, or both.

request/response header (RH). Control information associated with a particular RU. The RH precedes the request/response unit (RU) and specifies the type of RU (request unit or response unit).

request/response unit (RU). A generic term for a request unit or a response unit. See *request unit (RU)* and *response unit (RU)*.

requester. Synonym for *client*.

reset. (1) To cause a counter to take the state corresponding to a specified initial number. (I) (A) (2) To put all or part of a data processing device back into a prescribed state. (I) (A) (3) On a virtual circuit, reinitialization of data flow control. At reset, all data in transit are eliminated.

resource. (1) Any facility of a computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In the NetView program, any hardware or software that provides function to the network.

resource registration. The process of identifying names of resources, such as LUs, to a network node server or a central directory server.

resource sequence number (RSN). A value that identifies an update of a resource in a network topology database.

resource takeover. In VTAM, an action initiated by a network operator to transfer control of resources from one domain to another without breaking the connections or disrupting existing LU-LU sessions on the connection. See also *acquire* and *release*.

resource types. In the NetView program, a concept to describe the organization of panels. Resource types are defined as central processing unit, channel, control unit, and I/O device for one category; and communication controller, adapter, link, cluster controller, and terminal for another category. Resource types are combined with data types and display types to describe display organization. See also *data types* and *display types*.

response. (1) In data communication, a reply represented in the control field of a response frame. It

advises the primary or combined station of the action taken by the secondary or other combined station to one or more commands. (2) See also *command*.

response header (RH). (1) A header, optionally followed by a response unit (RU), that indicates whether the response is positive or negative and that may contain a pacing response. (2) See also *negative response*, *pacing response*, and *positive response*.

response unit (RU). A message unit that acknowledges a request unit. It may contain prefix information received in a request unit. If positive, the response unit may contain additional information (such as session parameters in response to BIND SESSION). If negative, the response unit contains sense data defining the exception condition.

return code. (1) A code used to influence the execution of succeeding instructions. (A) (2) A value returned to a program to indicate the results of an operation requested by that program.

REX. Route extension.

RH. Request/response header.

ring. See *ring network*.

ring network. (1) A network in which every node has exactly two branches connected to it and in which there are exactly two paths between any two nodes. (T) (2) A network configuration in which devices are connected by unidirectional transmission links to form a closed path.

ROM. Read-only memory. (A)

route. (1) An ordered sequence of nodes and transmission groups (TGs) that represent a path from an origin node to a destination node traversed by the traffic exchanged between them. (2) The path that network traffic uses to get from source to destination.

route addition resistance (RAR). A value that indicates a network node's capacity to perform intermediate session routing.

route daemon. A program that runs under 4BSD UNIX to propagate route information among machines on a local area network. Also referred to as *routed* (pronounced “route-d”).

route extension (REX). In SNA, the path control network components, including a peripheral link, that make up the portion of a path between a subarea node and a network addressable unit (NAU) in an adjacent peripheral node. See also *explicit route (ER)*, *path*, and *virtual route (VR)*.

Route Selection control vector (RSCV). A control vector that describes a route within an APPN network. The RSCV consists of an ordered sequence of control vectors that identify the TGs and nodes that make up the path from an origin node to a destination node.

routed. Pronounced “route-d.” See *route daemon*.

router. (1) A computer that determines the path of network traffic flow. The path selection is made from several paths based on information obtained from specific protocols, algorithms that attempt to identify the shortest or best path, and other criteria such as metrics or protocol-specific destination addresses. (2) An attaching device that connects two LAN segments, which use similar or different architectures, at the reference model network layer. (3) In OSI terminology, a function that determines a path by which an entity can be reached. (4) In TCP/IP, synonymous with *gateway*. (5) Contrast with *bridge*.

routine. A program, or part of a program, that may have some general or frequent use. (T)

routing. (1) The process of determining the path to be used for transmission of a message over a network. (T) (2) The assignment of the path by which a message is to reach its destination. (3) In SNA, the forwarding of a message unit along a particular path through a network, as determined by parameters carried in the message unit, such as the destination network address in a transmission header.

RouTing update Protocol (RTP). The VIrtual NEtworking System (VINES) protocol that maintains the routing database and allows the exchange of routing information between VINES nodes. See also *Internet Control Protocol (ICP)*.

RQD. Request discontact.

RSCV. Route Selection control vector.

RSN. Resource sequence number.

RTP. (1) Rapid Transport Protocol. (2) RouTing update Protocol.

RTP connection. See *Rapid Transport Protocol (RTP) connection*.

RU. Request/response unit.

RU chain. In SNA, a set of related request/response units (RUs) that are consecutively transmitted on a particular normal or expedited data flow. The request RU chain is the unit of recovery: if one of the RUs in the chain cannot be processed, the entire chain is discarded. Each RU belongs to only one chain, which has a beginning and an end indicated by means of control bits in request/response headers within the RU

chain. Each RU can be designated as first-in-chain (FIC), last-in-chain (LIC), middle-in-chain (MIC), or only-in-chain (OIC). Response units and expedited-flow request units are always sent as only-in-chain.

S

SAP. (1) Service access point. (2) Service Advertising Protocol.

SBCS. Single-byte character set.

SC. Session control.

SDLC. Synchronous Data Link Control.

secondary logical unit (SLU). In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. An LU may contain secondary and primary half-sessions for different active LU-LU sessions. Contrast with *primary logical unit (PLU)*.

secondary logical unit (SLU) key. A key-encrypting key used to protect a session cryptography key during its transmission to the secondary half-session.

segment. (1) A portion of a computer program that may be executed without the entire computer program being resident in main storage. (T) (2) A group of display elements. (3) A section of cable between components or devices. A segment may consist of a single patch cable, several patch cables that are connected, or a combination of building cable and patch cables that are connected. (4) In Internet communications, the unit of transfer between TCP functions in different machines. Each segment contains control and data fields; the current byte-stream position and actual data bytes are identified along with a checksum to validate received data. (5) In IMS/VS, the unit of access to a database; for the database system, the smallest amount of data that can be transferred by one DL/I operation. For input terminal operations using the DC feature, a segment is defined by the particular terminal type and is obtained by the application program with one call. (6) In the Enhanced X-Windows Toolkit, one or more lines that are drawn but not necessarily connected at the endpoints. (7) In LANs or WANs, a subset of nodes in a network or subnet that are connected by a common physical medium. (8) Synonym for *BIU segment*. (9) See also *link connection segment*.

select. To explicitly identify one or more objects to which a subsequent choice will apply.

selection. The process of explicitly identifying one or more objects to which a subsequent choice will apply.

send pacing. In SNA, the pacing of message units that a component is sending. Contrast with *receive pacing*.

sequence number. (1) In communications, a number assigned to a particular frame or packet to control the transmission flow and receipt of data. (2) A numerical value assigned by VTAM to each message exchanged between two nodes. The value (one for messages sent from the application program to the logical unit and another for messages sent from the logical unit to the application program) increases by one for each successive message transmitted unless it is reset by the application program with a set and test sequence numbers (STSN) indicator.

serial. (1) Pertaining to a process in which all events occur one after the other; for example, serial transmission of the bits of a character according to V24 CCITT protocol. (T) (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel. (A) (3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (A) (4) Contrast with *parallel*.

server. (1) A functional unit that provides shared services to workstations over a network; for example, a file server, a print server, a mail server. (T) (2) In a network, a data station that provides facilities to other stations; for example, a file server, a print server, a mail server. (A) (3) In the AIX operating system, an application program that usually runs in the background and is controlled by the system program controller. (4) In the Enhanced X-Windows Toolkit, a program that provides the basic windowing mechanism. It handles interprocess communication (IPC) connections from clients, demultiplexes graphics requests onto screens, and multiplexes input back to clients.

service access point (SAP). (1) In Open Systems Interconnection (OSI) architecture, the point at which the services of a layer are provided by an entity of that layer to an entity of the next higher layer. (T) (2) A logical point made available by an adapter where information can be received and transmitted. A single service access point can have many links terminating in it. (3) A logical address that allows a system to route data between a remote device and the appropriate communications support. (4) The identification of the services provided by a specific communication service provider to one of its users. For example, the Internet Protocol (IP) uses the services of a token-ring adapter. The service access point, in this case, is the name by which IP knows the adapter that is the token-ring address.

Service Advertising Protocol (SAP). In Internetwork Packet Exchange (IPX), a protocol that provides the following:

A mechanism that allows IPX servers on an internet to advertise their services by name and type. Servers using this protocol have their name, service type, and IP address recorded in all file servers running NetWare.

A mechanism that allows a workstation to broadcast a query to discover the identities of all servers of all types, all servers of a specific type, or the nearest server of a specific type.

A mechanism that allows a workstation to query any file server running NetWare to discover the names and addresses of all servers of a specific type.

service key. In Communications Manager/2, one of two access passwords (the other is the master key) used to unlock the keylock for Communications Manager/2.

service point command facility (SPCF). A program or function that exchanges data and control between the network operator, the link connection component manager (LCCM), and the link connection subsystem manager (LCSM).

session. (1) In network architecture, for the purpose of data communication between functional units, all the activities which take place during the establishment, maintenance, and release of the connection. (T) (2) A logical connection between two network accessible units (NAUs) that can be activated, tailored to provide various protocols, and deactivated, as requested. Each session is uniquely identified in a transmission header (TH) accompanying any transmissions exchanged during the session.

session activation request. In SNA, a request that activates a session between two network accessible units (NAUs) and specifies session parameters that control various protocols during session activity; for example, BIND and ACTPU. Contrast with *session deactivation request*.

session control (SC). In SNA, either of the following:

One of the components of transmission control. Session control is used to purge data flowing in a session after an unrecoverable error occurs, to resynchronize the data flow after such an error, and to perform cryptographic verification.

A request unit (RU) category used for requests and responses exchanged between the session control components of a session and for session activation and deactivation requests and responses.

session data. Data about a session, collected by the NetView program, that consists of session awareness data, session trace data, and session response time data.

session deactivation request. In SNA, a request that deactivates a session between two network accessible units (NAUs); for example, UNBIND and DACTPU. Synonymous with *generic unbind*. Contrast with *session activation request*.

session-level pacing. (1) A flow control technique that permits a receiving half-session or session connector to control the data transfer rate (the rate at which it receives request units) on the normal flow. It is used to prevent overloading a receiver with unprocessed requests when the sender can generate requests faster than the receiver can process them. (2) See also *adaptive session-level pacing*, *fixed session-level pacing*, and *virtual route (VR) pacing*.

session-level security. For logical unit (LU) 6.2, partner LU verification and session cryptography. See *conversation-level security*.

session limit. (1) The maximum number of concurrently active LU-LU sessions that a particular logical unit (LU) can support. (2) In NCP, the maximum number of concurrent line-scheduling sessions on a non-SDLC, multipoint line.

shared. Pertaining to the availability of a resource for more than one use at the same time.

shift-out character (SO). A code extension character that substitutes for the graphic characters of the standard character set an alternative set of graphic characters upon which an agreement has been arrived at or that has been designated using code extension procedures. (I) (A)

short name. In Communications Manager/2, the one-letter name (A through Z) of the presentation space or emulation session. Synonymous with *presentation space ID (PSID)* and *short-session ID*.

short-session ID. In Communications Manager/2, synonym for *short name*.

shutdown. The process of ending operation of a system or a subsystem, following a defined procedure.

Simple Network Management Protocol (SNMP). In the Internet suite of protocols, a network management protocol that is used to monitor routers and attached networks. SNMP is an application layer protocol. Information on devices managed is defined and stored in the application's Management Information Base (MIB).

single-byte character set (SBCS). A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set (DBCS)*.

SLU. Secondary logical unit.

SNA. Systems Network Architecture.

SNA management services (SNA/MS). The services provided to assist in management of SNA networks.

SNA network. The part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network accessible units (NAUs), boundary function, gateway function, and intermediate session routing function components; and the transport network.

SNAP. Subnetwork Access Protocol.

SNMP. Simple Network Management Protocol.

SO. The shift-out character. (I) (A)

socket. (1) An endpoint for communication between processes or application programs. (2) Synonym for *port*.

span. In the NetView program, a user-defined group of network resources within a single domain. Each major or minor node is defined as belonging to one or more spans. See also *span of control*.

span of control. The total network resources over which a particular network operator has control. All the network resources listed in spans associated through profile definition with a particular network operator are within that operator's span of control.

SPCF. Service point command facility.

specific-mode. In VTAM, the following:

The form of a RECEIVE request that obtains input from one specific session

The form of an ACCEPT request that completes the establishment of a session by accepting a specific queued CINIT request.

Contrast with *any-mode*. See *continue-specific mode*.

sphere of control (SOC). The set of control point domains served by a single management services focal point.

SPM. Sync point manager.

SSCP. System services control point.

SSCP-dependent LU. An LU that requires assistance from a system services control point (SSCP) in order to initiate an LU-LU session. It requires an SSCP-LU session.

SSCP-LU session. In SNA, a session between a system services control point (SSCP) and a logical unit (LU). The session enables the LU to request the SSCP to help initiate LU-LU sessions.

SSCP-PU session. In SNA, a session between a system services control point (SSCP) and a physical unit (PU); SSCP-PU sessions allow SSCPs to send requests to and receive status information from individual nodes in order to control the network configuration.

stack. Synonym for *pushdown list*.

statement. A language syntactic unit consisting of an operator, or other statement identifier, followed by one or more operands. See *definition statement*.

static. (1) In programming languages, pertaining to properties that can be established before execution of a program; for example, the length of a fixed length variable is static. (I) (2) Pertaining to an operation that occurs at a predetermined or fixed time. (3) Contrast with *dynamic*.

station. An input or output point of a system that uses telecommunication facilities; for example, one or more systems, computers, terminals, devices, and associated programs at a particular location that can send or receive data over a telecommunication line.

status. The condition or state of hardware or software, usually represented by a status code.

storage. (1) A functional unit into which data can be placed, in which they can be retained and from which they can be retrieved. (T) (2) The action of placing data into a storage device. (I) (A) (3) A storage device. (A)

stub. (1) A program module that transfers remote procedure calls (RPCs) and responses between a client and a server. Stubs perform marshalling, unmarshalling, and data format conversion. Both clients and servers have stubs. The Network Interface Definition Language (NIDL) compiler generates client and server stub code from an interface definition. (2) Hooking functions used as extensions to the protocol to generate protocol requests for the Enhanced X-Windows Toolkit. (3) A small module, link-edited into application code, that locates and transfers control to a larger body of related code.

subarea. A portion of the SNA network consisting of a subarea node, attached peripheral nodes, and associated resources. Within a subarea node, all network accessible units (NAUs), links, and adjacent link stations (in attached peripheral or subarea nodes) that are addressable within the subarea share a

common subarea address and have distinct element addresses.

subarea node (SN). A node that uses network addresses for routing and maintains routing tables that reflect the configuration of the network. Subarea nodes can provide gateway function to connect multiple subarea networks, intermediate routing function, and boundary function support for peripheral nodes. Type 4 and type 5 nodes can be subarea nodes.

subarea PU. In SNA, a physical unit in a subarea node. Contrast with *peripheral PU*.

subdirectory. A directory contained within another directory in a file system hierarchy.

Subnetwork Access Protocol (SNAP). In LANs, a 5-byte protocol discriminator that identifies the non-IEEE standard protocol family to which a packet belongs. The SNAP value is used to differentiate between protocols that use $AA as their service access point (SAP) value.

subsystem. A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system. (T)

subvector. A subcomponent of the network management vector transport (NMVT) major vector.

switched network. Any network in which connections are established by closing switches, for example, by dialing.

sync point. An intermediate or end point during processing of a transaction at which an update or modification to one or more of the transaction's protected resources is logically complete and error free. Synonymous with *synchronization point*.

sync point manager (SPM). The component of the node that implements two-phase commit and resynchronization processing. The subcomponents of the SPM are sync point services (SPS) and the protection managers (the conversation resource protection managers and the local resource protection managers).

synchronization point. Synonym for *sync point*.

synchronous. (1) Pertaining to two or more processes that depend upon the occurrence of specific events such as common timing signals. (T) (2) Occurring with a regular or predictable time relationship.

Synchronous Data Link Control (SDLC). (1) A discipline conforming to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute (ANSI) and High-level Data Link Control (HDLC) of the International Organization for Standardization, for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half-duplex over switched or nonswitched links. The configuration of the link connection may be point-to-point, multipoint, or loop. (I) (2) Contrast with *binary synchronous communication (BSC)*.

synchronous operation. In VTAM, a communication, or other operation in which VTAM, after receiving the request for the operation, does not return control to the program until the operation is completed. Contrast with *asynchronous operation*.

synchronous request. In VTAM, a request for a synchronous operation. Contrast with *asynchronous request*.

SYNTAX. In the Simple Network Management Protocol (SNMP), a clause in the MIB module that defines the abstract data structure that corresponds to a managed object.

system. In data processing, a collection of people, machines, and methods organized to accomplish a set of specific functions. (I) (A)

system restart. Synonym for *initial program load (IPL)*.

system services control point (SSCP). A component within a subarea network for managing the configuration, coordinating network operator and problem determination requests, and providing directory services and other session services for users of the network. Multiple SSCPs, cooperating as peers with one another, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its own domain.

system services control point (SSCP) domain. The system services control point, the physical units (PUs), the logical units (LUs), the links, the link stations, and all the resources that the SSCP has the ability to control by means of activation and deactivation requests.

system startup. Synonym for *initial program load (IPL)*.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. The layered structure of SNA allows the ultimate origins and destinations of information, that is, the users, to be independent of and

unaffected by the specific SNA network services and facilities used for information exchange.

T

table. A repository for data that NetDA/2 uses to design a network. Each table contains information related to the network.

task. In a multiprogramming or multiprocessing environment, one or more sequences of instructions treated by a control program as an element of work to be accomplished by a computer. (I) (A)

task panel. Online display from which you communicate with the program in order to accomplish the program's function, either by selecting an option provided on the panel or by entering an explicit command. See also *help panel*.

TERMINATE. In SNA, a request unit that is sent by a logical unit (LU) to its system services control point (SSCP) to cause the SSCP to start a procedure to end one or more designated LU-LU sessions.

TG. Transmission group.

TG vector. See *transmission group vector*.

TH. Transmission header.

thread. In the OS/2 operating system, the smallest unit of operation to be performed within a process.

threshold. (1) In the NetView program, a percentage value, set for a resource and compared to a calculated error-to-traffic ratio. (2) In NetView for AIX, a setting that specifies the maximum value a statistic can reach before notification that the limit was exceeded. For example, when a monitored MIB value has exceeded the threshold, the data collector generates a threshold event. (3) In NPM, high or low values supplied by the user to monitor data and statistics being collected. (4) In IBM bridge programs, a value set for the maximum number of frames that are not forwarded across a bridge due to errors, before a “threshold exceeded” occurrence is counted and indicated to network management programs. (5) An initial value from which a counter is decremented to 0, or a value to which a counter is incremented or decremented from an initial value.

time stamp. (1) To apply the current system time. (2) The value on an object that is an indication of the system time at some critical point in the history of the object. (3) In query, the identification of the day and time when a query report was created that query automatically provides on each report.

timeout. (1) An event that occurs at the end of a predetermined period of time that began at the occurrence of another specified event. (I) (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

token. (1) In a local area network, the symbol of authority passed successively from one data station to another to indicate the station temporarily in control of the transmission medium. Each data station has an opportunity to acquire and use the token to control the medium. A token is a particular message or bit pattern that signifies permission to transmit. (T) (2) In LANs, a sequence of bits passed from one device to another along the transmission medium. When the token has data appended to it, it becomes a frame.

token ring. (1) According to IEEE 802.5, network technology that controls media access by passing a token (special packet or frame) between media-attached stations. (2) A FDDI or IEEE 802.5 network with a ring topology that passes tokens from one attaching ring station (node) to another. (3) See also *local area network (LAN)*.

topology. In communications, the physical or logical arrangement of nodes in a network, especially the relationships among nodes and the links between them.

topology and routing services (TRS). An APPN control point component that manages the topology database, computes routes, and provides a Route Selection control vector (RSCV) that specifies the best route through the network for a given session based on its requested class of service.

topology database. See *local topology database* and *network topology database*.

TP. (1) Transaction program. (2) Transmission priority.

TR. Trace.

transaction. In IMS/VS, a specific set of input data that triggers execution of a specific process or job. A transaction is a message destined for an application program.

transaction program (TP). (1) A program that processes transactions in an SNA network. There are two kinds of transaction programs: application transaction programs and service transaction programs. See also *conversation*. (2) In VTAM, a program that performs services related to the processing of a transaction. One or more transaction programs may operate within a VTAM application program that is using the VTAM application program interface (API). In that situation, the transaction program would request

services from the application program, using protocols defined by that application program. The application program, in turn, could request services from VTAM by issuing the APPCCMD macroinstruction.

transmission group (TG). (1) A connection between adjacent nodes that is identified by a transmission group number. (2) In a subarea network, a single link or a group of links between adjacent nodes. When a transmission group consists of a group of links, the links are viewed as a single logical link, and the transmission group is called a *multilink transmission group (MLTG)*. A *mixed-media multilink transmission group (MMMLTG)* is one that contains links of different medium types (for example, token-ring, switched SDLC, nonswitched SDLC, and frame-relay links). (3) In an APPN network, a single link between adjacent nodes. (4) See also *parallel transmission groups*.

transmission group (TG) profile. In VTAM, a named set of characteristics (such as cost per byte, cost per unit of time, and capacity) that is used for APPN links.

transmission group (TG) vector. A representation of an endpoint TG in a T2.1 network, consisting of two control vectors: the TG Descriptor (X'46') control vector and the TG Characteristics (X'47') control vector.

transmission header (TH). Control information, optionally followed by a basic information unit (BIU) or a BIU segment, that is created and used by path control to route message units and to control their flow within the network. See also *path information unit*.

transmission priority. A rank assigned to a message unit that determines its precedence for being selected by the path control component in each node along a route for forwarding to the next node in the route.

transport protocol. A specification of the rules governing the exchange of information between components of a transport network.

trap. In the Simple Network Management Protocol (SNMP), a message sent by a managed node (agent function) to a management station to report an exception condition.

tutorial. Information presented in a teaching format.

T1. In the United States, a 1.544-Mbps public access line. It is available in twenty-four 64-Kbps channels. The European version (E1) transmits 2.048 Mbps. The Japanese version (J1) transmits 1.544 Mbps.

U

UNBIND. In SNA, a request to deactivate a session between two logical units (LUs). See also *session deactivation request*. Contrast with *BIND*.

uninterpreted name. In SNA, a character string that a system services control point (SSCP) can convert into the network name of a logical unit (LU). Typically, an uninterpreted name is used in a logon or Initiate request from a secondary logical unit (SLU) to identify the primary logical unit (PLU) with which the session is requested.

UP. Unnumbered poll.

upstream. In the direction of data flow from the user to the host. Contrast with *downstream*.

uptime. (1) Synonym for *operable time*. (T) (2) Deprecated term for *available time*. (3) Synonym for *operating time*.

user. (1) Any person or any thing that may issue or receive commands and messages to or from the information processing system. (T) (2) Anyone who requires the services of a computing system.

using node. (1) In NCP, the NCP in the host's domain that reports a link error condition. (2) For the command facility of the NetView program and for NCCF, the ID parameter of certain network control commands.

UTC. Coordinated universal time.

V

value. (1) A specific occurrence of an attribute; for example, “blue” for the attribute “color.” (T) (2) A quantity assigned to a constant, a variable, a parameter, or a symbol.

variable. (1) In programming languages, a language object that may take different values, one at a time. The values of a variable are usually restricted to a certain data type. (I) (2) A quantity that can assume any of a given set of values. (A) (3) A name used to represent a data item whose value can be changed while the program is running. (4) In the Simple Network Management Protocol (SNMP), a match of an object instance name with an associated value. (5) In the NetView command list language, a character string beginning with “&” that is coded in a command list and is assigned a value during execution of the command list.

vector. The MAC frame information field.

verb. See *LU 6.2 verb*.

version. A separately licensed program that usually has significant new code or new function.

viewing filter. In the NetView program, the function that allows a user to select the alert data to be displayed on a terminal. All other stored data is blocked.

virtual circuit. (1) In packet switching, the facilities provided by a network that give the appearance to the user of an actual connection. (T) See also *data circuit*. Contrast with *physical circuit*. (2) A logical connection established between two DTEs.

virtual node. Synonym for *virtual routing node*.

virtual route (VR). (1) In SNA, either (a) a logical connection between two subarea nodes that is physically realized as a particular explicit route or (b) a logical connection that is contained wholly within a subarea node for intranode sessions. A virtual route between distinct subarea nodes imposes a transmission priority on the underlying explicit route, provides flow control through virtual route pacing, and provides data integrity through sequence numbering of path information units (PIUs). (2) Contrast with *explicit route (ER)*. See also *path* and *route extension (REX)*.

virtual route (VR) pacing. In SNA, a flow control technique used by the virtual route control component of path control at each end of a virtual route to control the rate at which path information units (PIUs) flow over the virtual route. VR pacing can be adjusted according to traffic congestion in any of the nodes along the route. See also *session-level pacing*.

virtual routing node. A representation of a node's connectivity to a connection network defined on a shared-access transport facility, such as a token ring. Synonymous with *virtual node*.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VR. Virtual route.

VTAM. (1) Virtual Telecommunications Access Method. (2) Synonymous with *ACF/VTAM*.

VTAM operator command. A command used to monitor or control a VTAM domain. See also *definition statement*.

W

WAN. Wide area network.

weight. For route selection purposes, the degree to which resources (such as nodes and transmission groups) meet the criteria specified by a particular class of service. In APPN route selection, routes of minimum weight are chosen.

wide area network (WAN). (1) A network that provides communication services to a geographic area larger than that served by a local area network or a metropolitan area network, and that may use or provide public communication facilities. (T) (2) A data communication network designed to serve an area of hundreds or thousands of miles; for example, public and private packet-switching networks, and national telephone networks. (3) Contrast with *local area network (LAN)* and *metropolitan area network (MAN)*.

window. (1) A portion of a display surface in which display images pertaining to a particular application can be presented. Different applications can be displayed simultaneously in different windows. (A) (2) An area with visible boundaries that presents a view of an object or with which a user conducts a dialog with a computer system. (3) In data communication, the number of data packets a data terminal equipment (DTE) or data circuit-terminating equipment (DCE) can send across a logical channel before waiting for authorization to send another data packet. The window is the main mechanism of pacing, or flow control, of packets. (4) See *pacing window*.

window size. The specified number of frames of information that can be sent before receiving an acknowledgment response.

WinSock application programming interface (API). A socket-style transport interface developed for the Windows family of operating systems.

working directory. The directory to which an application program will write data if no path is specified.

workstation. (1) A functional unit at which a user works. A workstation often has some processing capability. (T) (2) One or more programmable or nonprogrammable devices that allow a user to do work. (3) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

X

XID. Exchange identification.

XMIT. Transmit.

Z

Z time. Abbreviation for *Zulu time*. Synonym for *coordinated universal time (UTC)*.

Zulu time (Z). Synonym for *coordinated universal time (UTC)*.

Index

A

ACTIVATE_SESSION 133
activation and deactivation verbs 11
  ACTIVATE_SESSION 133
  DEACTIVATE_CONV_GROUP 136
  DEACTIVATE_SESSION 138
  PATH_SWITCH 141
  START_DLC 116
  START_INTERNAL_PU 118
  START_LS 121
  START_PORT 123
  STOP_DLC 125
  STOP_INTERNAL_PU 127
  STOP_LS 129
  STOP_PORT 131
ALERT_INDICATION 398
 alerts, unsolicited 390
APING 356
APING verbs
  APING 356
Attach Manager verbs
  DISABLE_ATTACH_MANAGER 370
  ENABLE_ATTACH_MANAGER 371
  QUERY_ATTACH_MANAGER 372

B

buffer space required 12

C

CHANGE_SESSION_LIMIT 302
children 28
common VCB fields 10, 377
compiling and linking a Node Operator Facility program 6, 376
configuring the node for LEN support 15
Connection Network 15, 144
CPI-C verbs
  DEFINE_CPIC_ SIDE_INFO 361
  DELETE_CPIC_SIDE_INFO 364
  QUERY_CPIC_SIDE_INFO 366

D

data_lost indicator 14
DEACTIVATE_CONV_GROUP 136
DEACTIVATE_SESSION 138
DEFINE_ADJACENT_LEN_NODE 28, 88
DEFINE_CN 31
DEFINE_COS 35
DEFINE_CPIC_SIDE_INFO 361
DEFINE_DEFAULT_PU 41, 43
DEFINE_DLC 45
DEFINE_DLUR_DEFAULTS 48
DEFINE_FOCAL_POINT 50
DEFINE_INTERNAL_PU 54
DEFINE_LOCAL_LU 57
DEFINE_LS 60
DEFINE_LU_0_TO_3 70
DEFINE_LU_LU_PASSWORD 346
DEFINE_MODE 73
DEFINE_PARTNER_LU 76
DEFINE_PORT 79
DEFINE_TP 85
DEFINE_USERID_PASSWORD 348
DELETE_CN 90
DELETE_COS 92
DELETE_CPIC_SIDE_INFO 364
DELETE_DLC 94
DELETE_FOCAL_POINT 96
DELETE_INTERNAL_PU 98
DELETE_LOCAL_LU 100
DELETE_LS 102
DELETE_LU_0_TO_3 104
DELETE_LU_LU_PASSWORD 350
DELETE_MODE 106
DELETE_PARTNER_LU 108
DELETE_PORT 110
DELETE_TP 112
DELETE_USERID_PASSWORD 352
detailed information 13
DISABLE_ATTACH_MANAGER 370
DLC processes 15
DLC_INDICATION 314
DLL (dynamic link library) 376, 382
DLUS_INDICATION 315
double-byte character set support x
dynamic link library (DLL)
  See DLL (dynamic link library)
dynamically linking to a DLL 376

E

ENABLE_ATTACH_MANAGER 371
entry points
  for Management Services verbs
    WinMS() 380
    WinMSCleanup() 381
    WinMSRegisterApplication() 383
    WinMSStartup() 382
    WinMSUnregisterApplication() 385
  for Node Operator Facility verbs
    WinAsyncNOF() 19

© Copyright IBM Corp. 1989, 1996

entry points *(continued)*
for Node Operator Facility verbs *(continued)*
WinAsyncNOFEx() 20
WinNOF() 18
WinNOFCancelAsyncRequest() 21
WinNOFCleanup() 22
WinNOFGetIndication() 14, 26, 387
WinNOFRegisterIndicationSink() 14, 24
WinNOFStartup() 23
WinNOFUnregisterIndicationSink() 14, 25
introduction 5, 375
linkage information 377

F

focal point
domain 50
explicit 50
host 50
implicit backup 50
implicit primary 50
FOCAL_POINT_INDICATION 317
FP_INDICATION 399

G

general protection fault 6, 376

H

highlighting, how used in this book x
HPR (high-performance routing) 141

I

indication verbs
DLC_INDICATION 314
DLUS_INDICATION 315
FOCAL_POINT_INDICATION 317
LOCAL_TOPOLOGY_INDICATION 319
LS_INDICATION 320
LU_0_TO_3_INDICATION 324
MODE_INDICATION 328
PLU_INDICATION 330
PORT_INDICATION 332
PU_INDICATION 333
REGISTRATION_FAILURE 336
RTP_INDICATION 337
SESSION_INDICATION 341
INITIALIZE_SESSION_LIMIT 306

L

LEN node 15
limited resource 65
link stations
defined link stations 16
dynamic link stations 16
implicit link stations 16
temporary link stations 16
list_options field 12
AP_FIRST_IN_LIST 12
AP_LIST_FROM_NEXT 12
AP_LIST_INCLUSIVE 12
filtering options 12
index value 12
local descriptor table 6, 375
LOCAL_TOPOLOGY_INDICATION 319
LS_INDICATION 320
LU pool 70
LU_0_TO_3_INDICATION 324

M

management services verbs
ALERT_INDICATION 398
FP_INDICATION 399
MDS_MU_RECEIVED 393
NMVT_RECEIVED 400
SEND_MSD_MU 395
TRANSFER_MS_DATA 390
MDS_MU_RECEIVED 393
MODE_INDICATION 328

N

NMVT_RECEIVED 400
node configuration verbs
DEFINE_ADJACENT_LEN_NODE 28
DEFINE_CN 31
DEFINE_COS 35
DEFINE_DEFAULT_PU 43
DEFINE_DEFAULTS 41
DEFINE_DLC 45
DEFINE_DLUR_DEFAULTS 48
DEFINE_FOCAL_POINT 50
DEFINE_INTERNAL_PU 54
DEFINE_LOCAL_LU 57
DEFINE_LS 60
DEFINE_LU_0_TO_3 70
DEFINE_MODE 73
DEFINE_PARTNER_LU 76
DEFINE_PORT 79
DEFINE_TP 85
DELETE_ADJACENT_LEN_NODE 88
DELETE_CN 90
DELETE_COS 92
DELETE_DLC 94
DELETE_FOCAL_POINT 96
DELETE_INTERNAL_PU 98
DELETE_LOCAL_LU 100
DELETE_LS 102

node configuration verbs *(continued)*
DELETE_LU_0_TO_3 104
DELETE_MODE 106
DELETE_PARTNER_LU 108
DELETE_PORT 110
DELETE_TP 112
node row (in a class of service definition) 35

P

PATH_SWITCH 141
Personal Communications Management Services API 375
Personal Communications Node Operator Facility API 5
PLU_INDICATION 330
PORT_INDICATION 332
ports 15
nonswitched ports 15
SATF ports 15
switched ports 15
pseudonym files
WINMS.H 377
WINNOF.H 7
PU_INDICATION 333

Q

query verbs 12
QUERY_CN 144
QUERY_CN_PORT 149
QUERY_COS 152
QUERY_DEFAULT_PU 156
QUERY_DEFAULTS 158
QUERY_DIRECTORY_LU 160
QUERY_DIRECTORY_STATS 164
QUERY_DLC 166
QUERY_DLUR_LU 171
QUERY_DLUR_PU 175
QUERY_DLUS 181
QUERY_FOCAL_POINT 186
QUERY_LOCAL_LU 191
QUERY_LOCAL_TOPOLOGY 196
QUERY_LS 201
QUERY_LU_0_TO_3 214
QUERY_MDS_APPLICATION 222
QUERY_MDS_STATISTICS 225
QUERY_MODE 228
QUERY_MODE_DEFINITION 234
QUERY_MODE_TO_COS_MAPPING 238
QUERY_NMVT_APPLICATION 241
QUERY_NODE 244
QUERY_PARTNER_LU 252
QUERY_PARTNER_LU_DEFINITION 258
QUERY_PORT 263
QUERY_PU 271
query verbs *(continued)*
QUERY_RTP_CONNECTION 276
QUERY_SESSION 283
QUERY_STATISTICS 290
QUERY_TP 292
QUERY_TP_DEFINITION 296
QUERY_ATTACH_MANAGER 372
QUERY_CN 144
QUERY_CN_PORT 149
QUERY_COS 152
QUERY_CPIC_SIDE_INFO 366
QUERY_DEFAULT_PU 156
QUERY_DEFAULTS 158
QUERY_DIRECTORY_LU 160
QUERY_DIRECTORY_STATS 164
QUERY_DLC 166
QUERY_DLUR_LU 171
QUERY_DLUR_PU 175
QUERY_DLUS 181
QUERY_FOCAL_POINT 186
QUERY_LOCAL_LU 191
QUERY_LOCAL_TOPOLOGY 196
QUERY_LS 201
QUERY_LU_0_TO_3 214
QUERY_MDS_APPLICATION 222
QUERY_MDS_STATISTICS 225
QUERY_MODE 228
QUERY_MODE_DEFINITION 234
QUERY_MODE_TO_COS_MAPPING 238
QUERY_NMVT_APPLICATION 241
QUERY_NODE 244
QUERY_PARTNER_LU 252
QUERY_PARTNER_LU_DEFINITION 258
QUERY_PORT 263
QUERY_PU 271
QUERY_RTP_CONNECTION 276
QUERY_SESSION 283
QUERY_STATISTICS 290
QUERY_TP 292
QUERY_TP_DEFINITION 296

R

REGISTRATION_FAILURE 336
RESET_SESSION_LIMIT 310
RTP_INDICATION 337

S

SATF (shared-access transport facility) 15
security verbs
DEFINE_LU_LU_PASSWORD 346
DEFINE_USERID_PASSWORD 348
DELETE_LU_LU_PASSWORD 350
DELETE_USERID_PASSWORD 352

SEND_MDS_MU 395
session limit verbs
CHANGE_SESSION_LIMIT 302
INITIALIZE_SESSION_LIMIT 306
RESET_SESSION_LIMIT 310
SESSION_INDICATION 341
START_DLC 116
START_INTERNAL_PU 118, 127
START_LS 121
START_PORT 123
STOP_DLC 125
STOP_INTERNAL_PU
STOP_LS 129
STOP_PORT 131
summary information 13

T

TG row (in a class of service definition) 35
TRANSFER_MS_DATA 390

U

unsolicited alerts 390

V

verb control block
common fields 10
introduction 5, 6, 375
verbs
activating and deactivating at link level 11
START_DLC 116
START_INTERNAL_PU 118
START_LS 121
START_PORT 123
STOP_DLC 125
STOP_INTERNAL_PU 127
STOP_LS 129
STOP_PORT 131
activating and deactivating at session level 11
ACTIVATE_SESSION 133
DEACTIVATE_CONV_GROUP 136
DEACTIVATE_SESSION 138
allowing a management application to"ping" a remote LU 14
APING 356
allowing CPI-C side information to be managed 14
DEFINE_CPIC_SIDE_INFO 361
DELETE_CPIC_SIDE_INFO 364
QUERY_CPIC_SIDE_INFO 366
changing the number of sessions 13
CHANGE_SESSION_LIMIT 302
INITIALIZE_SESSION_LIMIT 306
RESET_SESSION_LIMIT 310
controlling the Attach Manager
DISABLE_ATTACH_MANAGER 370
verbs *(continued)*
controlling the Attach Manager *(continued)*
ENABLE_ATTACH_MANAGER 371
QUERY_ATTACH_MANAGER 372
defining resources 10
DEFINE_ADJACENT_LEN_NODE 28
DEFINE_CN 31
DEFINE_COS 35
DEFINE_DEFAULT_PU 43
DEFINE_DEFAULTS 41
DEFINE_DLC 45
DEFINE_DLUR_DEFAULTS 48
DEFINE_FOCAL_POINT 50
DEFINE_INTERNAL_PU 54
DEFINE_LOCAL_LU 57
DEFINE_LS 60
DEFINE_LU_0_TO_3 70
DEFINE_MODE 73
DEFINE_PARTNER_LU 76
DEFINE_PORT 79
DEFINE_TP 85
deleting resources 11
DELETE_ADJACENT_LEN_NODE 88
DELETE_CN 90
DELETE_COS 92
DELETE_DLC 94
DELETE_FOCAL_POINT 96
DELETE_INTERNAL_PU 98
DELETE_LOCAL_LU 100
DELETE_LS 102
DELETE_LU_0_TO_3 104
DELETE_MODE 106
DELETE_PARTNER_LU 108
DELETE_PORT 110
DELETE_TP 112
description of, how to read 9
common VCB fields 10, 377
returned parameters 9
supplied parameters 9
forcing an RTP connection to switch paths 11
PATH_SWITCH 141
overview 9
providing security 14
DEFINE_LU_LU_PASSWORD 346
DEFINE_USERID_PASSWORD 348
DELETE_LU_LU_PASSWORD 350
DELETE_USERID_PASSWORD 352
reporting potential problems to Management Services focal points 375
ALERT_INDICATION 398
FP_INDICATION 399
MDS_MU_RECEIVED 393
NMVT_RECEIVED 400
SEND_MDS_MU 395
TRANSFER_MS_DATA 390
returning different levels of information
QUERY_DIRECTORY_LU 160

verbs *(continued)*
returning different levels of information *(continued)*
QUERY_DLC 166
QUERY_DLUR_LU 171
QUERY_DLUR_PU 175
QUERY_LOCAL_LU 191
QUERY_LOCAL_TOPOLOGY 196
QUERY_LS 201
QUERY_LU_0_TO_3 214
QUERY_MODE 228
QUERY_MODE_DEFINITION 234
QUERY_PARTNER_LU 252
QUERY_PARTNER_LU_DEFINITION 258
QUERY_PORT 263
QUERY_RTP_CONNECTION 276
QUERY_SESSION 283
QUERY_TP_DEFINITION 296
returning node information in named fields 12
QUERY_DEFAULT_PU 156
QUERY_DIRECTORY_STATS 164
QUERY_MDS_STATISTICS 225
QUERY_NODE 244
QUERY_STATISTICS 290
returning one of more units of information 12
QUERY_CN 144
QUERY_CN_PORT 149
QUERY_COS 152
QUERY_DEFAULTS 158
QUERY_DLUS 181
QUERY_FOCAL_POINT 186
QUERY_MDS_APPLICATION 222
QUERY_MODE_TO_COS_MAPPING 238
QUERY_NMVT_APPLICATION 241
QUERY_PU 271
QUERY_TP 292
QUERY_USERID_PASSWORD
summary 10
unsolicited indications of named events 13
DLC_INDICATION 314
DLUS_INDICATION 315
FOCAL_POINT_INDICATION 317
LOCAL_TOPOLOGY_INDICATIION 319
LS_INDICATION 320
LU_0_TO_3_INDICATION 324
MODE_INDICATION 328
PLU_INDICATION 330
PORT_INDICATION 332
PU_INDICATION 333
registering an application to receive information 13
REGISTRATION_FAILURE 336
RTP_INDICATION 337
SESSION_INDICATION 341
unregistering an application when it no longer requires information 13

W

wildcard entries 28, 88
WinAsyncNOF() 19
WinAsyncNOFEx() 20
WINMS.H file 377
WinMS() 380
WinMSCleanup() 381
WinMSRegisterApplication() 383
WinMSStartup() 382
WinMSUnregisterApplication() 385
WINNOF.H file 7
WinNOF() 18
WINNOF32.LIB file 6, 376
WinNOFCancelAsyncRequest() 21
WinNOFCleanup() 22
WinNOFGetIndication() 14, 26, 387
WinNOFRegisterIndicationSink() 14, 24
WinNOFStartup() 23
WinNOFUnregisterIndicationSink() 14, 25

X

XID 15
XID3 15

Communicating Your Comments to IBM

Personal Communications
AS/400 and 3270 for Windows** 95
System Management Programming
Version 4.1

Publication No. SC31-8199-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

If you prefer to send comments by mail, use the RCF at the back of this book.

If you prefer to send comments by FAX, use this number:

United States and Canada: **1-800-227-5088**

If you prefer to send comments electronically, use this network ID:

- IBM Mail Exchange: **USIB2HPD at IBMMAIL**
- IBMLink: **CIBMORCF at RALVM13**
- Internet: **USIB2HPD@VNET.IBM.COM**

Make sure to include the following in your note:

Title and publication number of this book
Page number or topic to which your comment applies.

Help us help you!

Personal Communications
AS/400 and 3270 for Windows 95**
System Management Programming
Version 4.1

Publication No. SC31-8199-00

We hope you find this publication useful, readable and technically accurate, but only you can tell us! Your comments and suggestions will help us improve our technical publications. Please take a few minutes to let us know what you think by completing this form.

Overall, how satisfied are you with the information in this book?	Satisfied	Dissatisfied

How satisfied are you that the information in this book is:	Satisfied	Dissatisfied
Accurate		
Complete		
Easy to find		
Easy to understand		
Well organized		
Applicable to your task		

Specific Comments or Problems:

Please tell us how we can improve this book:

Thank you for your response. When you send information to IBM, you grant IBM the right to use or distribute the information without incurring any obligation to you. You of course retain the right to use the information in any way you choose.

Your Internet Address: _____

Name

Address

Company or Organization

Phone No.

Help us help you!
SC31-8199-00

Cut or Fold
Along Line

IBM®

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Information Development
Department CGMD
International Business Machines Corporation
PO BOX 12195
RESEARCH TRIANGLE PARK NC 27709-9990

Fold and Tape **Please do not staple** Fold and Tape

SC31-8199-00

Cut or Fold
Along Line

IBM®

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-8199-

Artwork Definitions

id	File	Page	References
COV1	DZQL1SET	i	i
DZQL1C2	DZQL1SET	i	452

LERS Definitions

id	File	Page	References
DZQLER1	DZQL1SET	i	17, 27, 115, 143, 301, 313, 345, 355, 360, 369, 379, 389

Figures

id	File	Page	References
KEYEX1	DZQL1GLO	415	1 414
KEYEX2	DZQL1GLO	415	2 414
KEYEX3	DZQL1GLO	415	3 414

Headings

id	File	Page	References
NOTICES	DZQL1NOT	vii	Notices ii
TRADE	DZQL1NOT	viii	Trademarks
WHO	DZQL1PRE	ix	Who Should Read This Book
HOW	DZQL1PRE	ix	How to Use This Book
HIGH	DZQL1PRE	x	Conventions Used in This Book
REL	DZQL1PRE	xi	Related Publications
PART1	DZQL1MST SCRIPT	1	Part 1, Personal Communications Node Operator Facility ix
INTRO	DZQL1CH1	5	Chapter 1, Introduction ix
PUR	DZQL1CH1	5	Purpose of the Document
NOF	DZQL1CH1	5	Personal Communications Node Operator Facility
OVERV	DZQL1CH2	9	Chapter 2, Overview ix
VBOVW	DZQL1CH2	9	Overview of the Verbs in This Book
VDSUM	DZQL1CH2	10	Verb Summary
NODCFG	DZQL1CH2	10	Node Configuration
ACTDEA	DZQL1CH2	11	Activation and Deactivation
QNOD	DZQL1CH2	12	Querying the Node

144, 149, 152, 160, 166, 171, 175, 181, 186, 191, 196, 201, 222, 228, 234, 238, 241, 252, 258, 263, 271, 276, 283, 292, 296

CHNUM DZQL1CH2 13 Changing the number of sessions

UNSOL DZQL1CH2 13 Unsolicited Indications

SECVB DZQL1CH2 14 Security Verbs

APING DZQL1CH2 14 APING Verbs

CPIC DZQL1CH2 14 CPI-C Verbs

CFGLEN DZQL1CH2 15 Configuring the Node for LEN Node support

DLCPRO DZQL1CH2 15 DLC Processes, Ports, and Link Stations
45, 60, 79, 116, 121, 123

DPROC DZQL1CH2 15 DLC Processes

PORT DZQL1CH2 15 Ports

LINK DZQL1CH2 16 Link Stations

ENTRY DZQL1ENP 17 Chapter 3, Node Operator Facility Entry Points
ix, 6, 375

WINN1 DZQL1ENP 18 WinNOF()

WIN1 DZQL1ENP 19 WinAsyncNOF()

WIN2 DZQL1ENP 20 WinAsyncNOFEx()

WIN3 DZQL1ENP 21 WinNOFCancelAsyncRequest()

WIN4 DZQL1ENP 22 WinNOFCleanup()

WIN5 DZQL1ENP 23 WinNOFStartup()

WIN6 DZQL1ENP 24 WinNOFRegisterIndicationSink()

WIN7 DZQL1ENP 25 WinNOFUnregisterIndicationSink()

WIN8 DZQL1ENP 26 WinNOFGetIndication()

VERBS DZQL1NCF 27 Chapter 4, Node Configuration Verbs
ix, 9

DEFCN DZQL1NCF 31 DEFINE_CN
67, 200, 212

DEFCOS DZQL1NCF 35 DEFINE_COS
83, 269

DEFDLC DZQL1NCF 45 DEFINE_DLC
207

DEFFP DZQL1NCF 50 DEFINE_FOCAL_POINT
96

ACTDEAC DZQL1ACT 115 Chapter 5, Activation and Deactivation Verbs

QRY DZQL1QRY 143 Chapter 6, Query Verbs

LIMIT DZQL1SLM 301 Chapter 7, Session Limit Verbs

NOOP DZQL1IND 313 Chapter 8, Node Operator Facility API Indications

REGFAIL DZQL1IND 336 REGISTRATION_FAILURE
247

SECURE DZQL1SEC 345 Chapter 9, Security Verbs

ACPIC DZQL1CPI 355 Chapter 10, APING and CPI-C Verbs

ATM DZQL1ATM 369 Chapter 11, Attach Manager Verbs
ix, 9

PART2 DZQL1MST SCRIPT 373 Part 2, Personal Communications Management Services API
ix

INTROMS	DZQL1MSI	375	Chapter 12, Introduction to Management Services API ix
MGMTSV	DZQL1MSI	375	Management Services Verbs
ENTRYMS	DZQL1MSE	379	Chapter 13, Management Services Entry Points
WINMS1	DZQL1MSE	380	WinMS()
WINMS4	DZQL1MSE	381	WinMSCleanup()
WINMS5	DZQL1MSE	382	WinMSStartup()
WINMS6	DZQL1MSE	383	WinMSRegisterApplication()
WINMS7	DZQL1MSE	385	WinMSUnregisterApplication()
WINMS8	DZQL1MSE	387	WinMSGetIndication()
CH4MSV	DZQL1MGT	389	Chapter 14, Management Services Verbs ix, 43, 222, 223, 241, 242, 243
APA	DZQL1APA	403	Appendix A, IBM APPN MIB Tables 12

Index Entries

id	File	Page	References
PCNOF01	DZQL1INX	viii	(1) Personal Communications Node Operator Facility API 5
MS01	DZQL1INX	viii	(1) Personal Communications Management Services API 375
FUN01	DZQL1INX	viii	(1) verbs viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii
VBOVW	DZQL1INX	viii	(1) verbs (2) overview 9
VBSUM	DZQL1INX	viii	(1) verbs (2) summary 10
VBDESC	DZQL1INX	viii	(1) verbs (2) description of, how to read viii, viii, viii, 9
RETPARM	DZQL1INX	viii	(1) verbs (2) description of, how to read (3) returned parameters 9
SUPPARM	DZQL1INX	viii	(1) verbs (2) description of, how to read (3) supplied parameters 9
VBS0112	DZQL1INX	viii	(1) verbs (2) description of, how to read (3) common VCB fields 10, 377
FUN002	DZQL1INX	viii	(1) verbs (2) defining resources viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, 10
FUN0021	DZQL1INX	viii	(1) verbs (2) defining resources (3) DEFINE_ADJACENT_LEN_NODE 28
FUN0022	DZQL1INX	viii	(1) verbs

(2) defining resources
(3) DEFINE_CN
31

FUN0023 DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_COS
35

DEFDEFS DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_DEFAULTS
41

FUN0024 DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_DEFAULT_PU
43

FUN0025 DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_DLC
45

FUN0026 DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_DLUR_DEFAULTS
48

FUN0027 DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_FOCAL_POINT
50

FUN0028 DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_INTERNAL_PU
54

FUN0029 DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_LOCAL_LU
57

FUN002A DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_LS
60

FUN002B DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_LU_0_TO_3
70

FUN002E DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_MODE
73

FUN002F DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_PARTNER_LU
76

FUN002G DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_PORT
79

FUN002H DZQL1INX
viii (1) verbs
(2) defining resources
(3) DEFINE_TP
85

FUN003 DZQL1INX
viii (1) verbs
(2) deleting resources
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, 11

FUN0031 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_ADJACENT_LEN_NODE

88

FUN0032 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_CN
90

FUN0033 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_COS
92

FUN0034 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_DLC
94

FUN0035 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_FOCAL_POINT
96

FUN0036 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_INTERNAL_PU
98

FUN0037 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_LOCAL_LU
100

FUN0038 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_LS
102

FUN0039 DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_LU_0_TO_3
104

FUN003C DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_MODE
106

FUN003D DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_PARTNER_LU
108

FUN003E DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_PORT
110

FUN003F DZQL1INX
viii (1) verbs
(2) deleting resources
(3) DELETE_TP
112

FUN004 DZQL1INX
viii (1) verbs
(2) activating and deactivating at link level
viii, viii, viii, viii, viii, viii, viii, viii, 11

FUN0041 DZQL1INX
viii (1) verbs
(2) activating and deactivating at link level
(3) START_DLC
116

FUN0042 DZQL1INX
viii (1) verbs
(2) activating and deactivating at link level
(3) START_INTERNAL_PU
118

FUN0043 DZQL1INX
viii (1) verbs
(2) activating and deactivating at link level
(3) START_LS
121

FUN0044 DZQL1INX

viii (1) verbs
(2) activating and deactivating at link level
(3) START_PORT
123

FUN0045 DZQL1INX
viii (1) verbs
(2) activating and deactivating at link level
(3) STOP_DLC
125

FUN0046 DZQL1INX
viii (1) verbs
(2) activating and deactivating at link level
(3) STOP_INTERNAL_PU
127

FUN0047 DZQL1INX
viii (1) verbs
(2) activating and deactivating at link level
(3) STOP_LS
129

FUN0048 DZQL1INX
viii (1) verbs
(2) activating and deactivating at link level
(3) STOP_PORT
131

FUN005 DZQL1INX
viii (1) verbs
(2) activating and deactivating at session level
viii, viii, viii, 11

FUN0051 DZQL1INX
viii (1) verbs
(2) activating and deactivating at session level
(3) ACTIVATE_SESSION
133

FUN0052 DZQL1INX
viii (1) verbs
(2) activating and deactivating at session level
(3) DEACTIVATE_CONV_GROUP
136

FUN0053 DZQL1INX
viii (1) verbs
(2) activating and deactivating at session level
(3) DEACTIVATE_SESSION
138

FUN006 DZQL1INX
viii (1) verbs
(2) forcing an RTP connection to switch paths
viii, 11

FUN0061 DZQL1INX
viii (1) verbs
(2) forcing an RTP connection to switch paths
(3) PATH_SWITCH
141

FUN007 DZQL1INX
viii (1) verbs
(2) returning node information in named fields
viii, viii, viii, viii, viii, 12

FUN0071 DZQL1INX
viii (1) verbs
(2) returning node information in named fields
(3) QUERY_DEFAULT_PU
156

FUN0072 DZQL1INX
viii (1) verbs
(2) returning node information in named fields
(3) QUERY_DIRECTORY_STATS
164

FUN0073 DZQL1INX
viii (1) verbs
(2) returning node information in named fields
(3) QUERY_MDS_STATISTICS
225

FUN0074 DZQL1INX
viii (1) verbs
(2) returning node information in named fields
(3) QUERY_NODE
244

FUN0075 DZQL1INX
viii (1) verbs
(2) returning node information in named fields
(3) QUERY_STATISTICS
290

FUN008 DZQL1INX
viii (1) verbs
(2) returning one of more units of information
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, 12

FUN0082 DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_CN
144

FUN0083 DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_CN_PORT
149

FUN0084 DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_COS
152

FUN0085 DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_DEFAULTS
158

FUN0086 DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_DLUS
181

FUN0087 DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_FOCAL_POINT
186

FUN008A DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_MDS_APPLICATION
222

FUN008B DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_MODE_TO_COS_MAPPING
238

FUN008C DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_NMVT_APPLICATION
241

FUN008D DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_PU
271

FUN008E DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_TP
292

FUN008F DZQL1INX
viii (1) verbs
(2) returning one of more units of information
(3) QUERY_USERID_PASSWORD

FUN009 DZQL1INX
viii (1) verbs
(2) returning different levels of information
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii

FUN0091 DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_DIRECTORY_LU
160

FUN0092 DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_DLC
166

FUN0093 DZQL1INX
viii (1) verbs
(2) returning different levels of information

(3) QUERY_DLUR_LU
171
FUN0094 DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_DLUR_PU
175
FUN0095 DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_LOCAL_LU
191
FUN0096 DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_LOCAL_TOPOLOGY
196
FUN0097 DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_LS
201
FUN0098 DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_LU_0_TO_3
214
FUN0099 DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_MODE
228
FUN009A DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_MODE_DEFINITION
234
FUN009B DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_PARTNER_LU
252
FUN009C DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_PARTNER_LU_DEFINITION
258
FUN009D DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_PORT
263
FUN009E DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_RTP_CONNECTION
276
FUN009F DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_SESSION
283
FUN009G DZQL1INX
viii (1) verbs
(2) returning different levels of information
(3) QUERY_TP_DEFINITION
296
FUN010 DZQL1INX
viii (1) verbs
(2) changing the number of sessions
viii, viii, viii, 13
FUN0101 DZQL1INX
viii (1) verbs
(2) changing the number of sessions
(3) CHANGE_SESSION_LIMIT
302
FUN0102 DZQL1INX
viii (1) verbs
(2) changing the number of sessions
(3) INITIALIZE_SESSION_LIMIT
306

FUN0103 DZQL1INX
viii (1) verbs
(2) changing the number of sessions
(3) RESET_SESSION_LIMIT
310

FUN011 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, 13

FUNREG DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) registering an application to receive information
13

FUNUNRG DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) unregistering an application when it no longer requires information
13

FUN0111 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) DLC_INDICATION
314

FUN0112 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) DLUS_INDICATION
315

FUN0113 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) FOCAL_POINT_INDICATION
317

FUN0114 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) LOCAL_TOPOLOGY_INDICATIION
319

FUN0115 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) LS_INDICATION
320

FUN0116 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) LU_0_TO_3_INDICATION
324

FUN0117 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) MODE_INDICATION
328

FUN0118 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) PLU_INDICATION
330

FUN0119 DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) PORT_INDICATION
332

FUN011A DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) PU_INDICATION
333

FUN011B DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) REGISTRATION_FAILURE
336

FUN011C DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) RTP_INDICATION
337

FUN011D DZQL1INX
viii (1) verbs
(2) unsolicited indications of named events
(3) SESSION_INDICATION
341
FUN012 DZQL1INX
viii (1) verbs
(2) providing security
viii, viii, viii, viii, 14
FUN0121 DZQL1INX
viii (1) verbs
(2) providing security
(3) DEFINE_LU_LU_PASSWORD
346
FUN0122 DZQL1INX
viii (1) verbs
(2) providing security
(3) DEFINE_USERID_PASSWORD
348
FUN0123 DZQL1INX
viii (1) verbs
(2) providing security
(3) DELETE_LU_LU_PASSWORD
350
FUN0124 DZQL1INX
viii (1) verbs
(2) providing security
(3) DELETE_USERID_PASSWORD
352
FUN013 DZQL1INX
viii (1) verbs
(2) allowing a management application to"ping" a remote LU
viii, 14
FUN0131 DZQL1INX
viii (1) verbs
(2) allowing a management application to"ping" a remote LU
(3) APING
356
FUN014 DZQL1INX
viii (1) verbs
(2) allowing CPI-C side information to be managed
viii, viii, viii, 14
FUN0141 DZQL1INX
viii (1) verbs
(2) allowing CPI-C side information to be managed
(3) DEFINE_CPIC_SIDE_INFO
361
FUN0142 DZQL1INX
viii (1) verbs
(2) allowing CPI-C side information to be managed
(3) DELETE_CPIC_SIDE_INFO
364
FUN0143 DZQL1INX
viii (1) verbs
(2) allowing CPI-C side information to be managed
(3) QUERY_CPIC_SIDE_INFO
366
FUN015 DZQL1INX
viii (1) verbs
(2) reporting potential problems to Management Services focal points
viii, viii, viii, viii, viii, viii, 375
FUN0151 DZQL1INX
viii (1) verbs
(2) reporting potential problems to Management Services focal points
(3) ALERT_INDICATION
398
FUN0152 DZQL1INX
viii (1) verbs
(2) reporting potential problems to Management Services focal points
(3) FP_INDICATION
399
FUN0153 DZQL1INX
viii (1) verbs
(2) reporting potential problems to Management Services focal points
(3) MDS_MU_RECEIVED
393
FUN0154 DZQL1INX

viii (1) verbs
(2) reporting potential problems to Management Services focal points
(3) NMVT_RECEIVED
400

FUN0155 DZQL1INX
viii (1) verbs
(2) reporting potential problems to Management Services focal points
(3) SEND_MDS_MU
395

FUN0156 DZQL1INX
viii (1) verbs
(2) reporting potential problems to Management Services focal points
(3) TRANSFER_MS_DATA
390

FUN016 DZQL1INX
viii (1) verbs
(2) controlling the Attach Manager
viii, viii, viii

FUN0161 DZQL1INX
viii (1) verbs
(2) controlling the Attach Manager
(3) DISABLE_ATTACH_MANAGER
370

FUN0162 DZQL1INX
viii (1) verbs
(2) controlling the Attach Manager
(3) ENABLE_ATTACH_MANAGER
371

FUN0163 DZQL1INX
viii (1) verbs
(2) controlling the Attach Manager
(3) QUERY_ATTACH_MANAGER
372

LIS01 DZQL1INX
viii (1) **list_options** field
viii, viii, viii, viii, viii, 12

LISFILT DZQL1INX
viii (1) **list_options** field
(2) filtering options
12

LISIV DZQL1INX
viii (1) **list_options** field
(2) index value
12

LIS011 DZQL1INX
viii (1) **list_options** field
(2) AP_FIRST_IN_LIST
12

LIS012 DZQL1INX
viii (1) **list_options** field
(2) AP_LIST_INCLUSIVE
12

LIS013 DZQL1INX
viii (1) **list_options** field
(2) AP_LIST_FROM_NEXT
12

BUF01 DZQL1INX
viii (1) buffer space required
12

CON02 DZQL1INX
viii (1) configuring the node for LEN support
15

DLC01 DZQL1INX
viii (1) DLC processes
15

DLP01 DZQL1INX
viii (1) ports
viii, viii, viii, 15

DLP011 DZQL1INX
viii (1) ports
(2) switched ports
15

DLP012 DZQL1INX
viii (1) ports
(2) nonswitched ports
15

DLP013 DZQL1INX
viii (1) ports

(2) SATF ports
15

DLL01 DZQL1INX
viii (1) link stations
viii, viii, viii, viii

DLL011 DZQL1INX
viii (1) link stations
(2) defined link stations
16

DLL012 DZQL1INX
viii (1) link stations
(2) dynamic link stations
16

DLL013 DZQL1INX
viii (1) link stations
(2) implicit link stations
16

DLL014 DZQL1INX
viii (1) link stations
(2) temporary link stations
16

CVF01 DZQL1INX
viii (1) common VCB fields
10, 377

MSV01 DZQL1INX
viii (1) management services verbs
viii, viii, viii, viii, viii, viii

MSV011 DZQL1INX
viii (1) management services verbs
(2) ALERT_INDICATION
398

MSV012 DZQL1INX
viii (1) management services verbs
(2) FP_INDICATION
399

MSV013 DZQL1INX
viii (1) management services verbs
(2) MDS_MU_RECEIVED
393

MSV014 DZQL1INX
viii (1) management services verbs
(2) NMVT_RECEIVED
400

MSV015 DZQL1INX
viii (1) management services verbs
(2) SEND_MSD_MU
395

MSV016 DZQL1INX
viii (1) management services verbs
(2) TRANSFER_MS_DATA
390

NOD01 DZQL1INX
viii (1) node configuration verbs
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii,
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii

NOD013 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_ADJACENT_LEN_NODE
28

NOD014 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_CN
31

NOD015 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_COS
35

NFDFDF DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_DEFAULTS
41

NOD016 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_DEFAULT_PU
43

NOD017 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_DLC
45

NOD018 DZQL1INX
viii (1) node configuration verbs

(2) DEFINE_DLUR_DEFAULTS
48
NOD019 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_FOCAL_POINT
50
NOD020 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_INTERNAL_PU
54
NOD021 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_LOCAL_LU
57
NOD022 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_LS
60
NOD023 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_LU_0_TO_3
70
NOD026 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_MODE
73
NOD027 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_PARTNER_LU
76
NOD028 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_PORT
79
NOD029 DZQL1INX
viii (1) node configuration verbs
(2) DEFINE_TP
85
NOD030 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_ADJACENT_LEN_NODE
88
NOD031 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_CN
90
NOD032 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_COS
92
NOD033 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_DLC
94
NOD034 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_FOCAL_POINT
96
NOD035 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_INTERNAL_PU
98
NOD036 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_LOCAL_LU
100
NOD037 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_LS
102
NOD038 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_LU_0_TO_3
104
NOD041 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_MODE
106
NOD042 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_PARTNER_LU

108
NOD043 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_PORT
110
NOD044 DZQL1INX
viii (1) node configuration verbs
(2) DELETE_TP
112
ACD01 DZQL1INX
viii (1) activation and deactivation verbs
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, 11
ACD011 DZQL1INX
viii (1) activation and deactivation verbs
(2) START_DLC
116
ACD012 DZQL1INX
viii (1) activation and deactivation verbs
(2) START_INTERNAL_PU
118
ACD013 DZQL1INX
viii (1) activation and deactivation verbs
(2) START_LS
121
ACD014 DZQL1INX
viii (1) activation and deactivation verbs
(2) START_PORT
123
ACD015 DZQL1INX
viii (1) activation and deactivation verbs
(2) STOP_DLC
125
ACD016 DZQL1INX
viii (1) activation and deactivation verbs
(2) STOP_INTERNAL_PU
127
ACD017 DZQL1INX
viii (1) activation and deactivation verbs
(2) STOP_LS
129
ACD018 DZQL1INX
viii (1) activation and deactivation verbs
(2) STOP_PORT
131
ACD019 DZQL1INX
viii (1) activation and deactivation verbs
(2) ACTIVATE_SESSION
133
ACD020 DZQL1INX
viii (1) activation and deactivation verbs
(2) DEACTIVATE_CONV_GROUP
136
ACD021 DZQL1INX
viii (1) activation and deactivation verbs
(2) DEACTIVATE_SESSION
138
ACD022 DZQL1INX
viii (1) activation and deactivation verbs
(2) PATH_SWITCH
141
QUE01 DZQL1INX
viii (1) query verbs
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii,
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii,
viii, viii, viii, viii, 12
QUE012 DZQL1INX
viii (1) query verbs
(2) QUERY_CN
144
QUE013 DZQL1INX
viii (1) query verbs
(2) QUERY_CN_PORT
149
QUE014 DZQL1INX
viii (1) query verbs
(2) QUERY_COS
152
QUE015 DZQL1INX
viii (1) query verbs
(2) QUERY_DEFAULT_PU
156

QUE016 DZQL1INX
viii (1) query verbs
(2) QUERY_DEFAULTS
158
QUE017 DZQL1INX
viii (1) query verbs
(2) QUERY_DIRECTORY_LU
160
QUE018 DZQL1INX
viii (1) query verbs
(2) QUERY_DIRECTORY_STATS
164
QUE019 DZQL1INX
viii (1) query verbs
(2) QUERY_DLC
166
QUE020 DZQL1INX
viii (1) query verbs
(2) QUERY_DLUR_LU
171
QUE021 DZQL1INX
viii (1) query verbs
(2) QUERY_DLUR_PU
175
QUE022 DZQL1INX
viii (1) query verbs
(2) QUERY_DLUS
181
QUE023 DZQL1INX
viii (1) query verbs
(2) QUERY_FOCAL_POINT
186
QUE024 DZQL1INX
viii (1) query verbs
(2) QUERY_LOCAL_LU
191
QUE025 DZQL1INX
viii (1) query verbs
(2) QUERY_LOCAL_TOPOLOGY
196
QUE026 DZQL1INX
viii (1) query verbs
(2) QUERY_LS
201
QUE027 DZQL1INX
viii (1) query verbs
(2) QUERY_LU_0_TO_3
214
QUE029 DZQL1INX
viii (1) query verbs
(2) QUERY_MDS_APPLICATION
222
QUE030 DZQL1INX
viii (1) query verbs
(2) QUERY_MDS_STATISTICS
225
QUE031 DZQL1INX
viii (1) query verbs
(2) QUERY_MODE
228
QUE032 DZQL1INX
viii (1) query verbs
(2) QUERY_MODE_DEFINITION
234
QUE033 DZQL1INX
viii (1) query verbs
(2) QUERY_MODE_TO_COS_MAPPING
238
QUE034 DZQL1INX
viii (1) query verbs
(2) QUERY_NMVT_APPLICATION
241
QUE035 DZQL1INX
viii (1) query verbs
(2) QUERY_NODE
244
QUE036 DZQL1INX
viii (1) query verbs
(2) QUERY_PARTNER_LU
252
QUE037 DZQL1INX

viii (1) query verbs
(2) QUERY_PARTNER_LU_DEFINITION
258

QUE038 DZQL1INX
viii (1) query verbs
(2) QUERY_PORT
263

QUE039 DZQL1INX
viii (1) query verbs
(2) QUERY_PU
271

QUE040 DZQL1INX
viii (1) query verbs
(2) QUERY_RTP_CONNECTION
276

QUE041 DZQL1INX
viii (1) query verbs
(2) QUERY_SESSION
283

QUE042 DZQL1INX
viii (1) query verbs
(2) QUERY_STATISTICS
290

QUE043 DZQL1INX
viii (1) query verbs
(2) QUERY_TP
292

QUE044 DZQL1INX
viii (1) query verbs
(2) QUERY_TP_DEFINITION
296

SES01 DZQL1INX
viii (1) session limit verbs
viii, viii, viii

SES011 DZQL1INX
viii (1) session limit verbs
(2) CHANGE_SESSION_LIMIT
302

SES012 DZQL1INX
viii (1) session limit verbs
(2) INITIALIZE_SESSION_LIMIT
306

SES013 DZQL1INX
viii (1) session limit verbs
(2) RESET_SESSION_LIMIT
310

NOF01 DZQL1INX
viii (1) indication verbs
viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii, viii

NOF011 DZQL1INX
viii (1) indication verbs
(2) DLC_INDICATION
314

NOF012 DZQL1INX
viii (1) indication verbs
(2) DLUS_INDICATION
315

NOF013 DZQL1INX
viii (1) indication verbs
(2) FOCAL_POINT_INDICATION
317

NOF014 DZQL1INX
viii (1) indication verbs
(2) LOCAL_TOPOLOGY_INDICATION
319

NOF015 DZQL1INX
viii (1) indication verbs
(2) LS_INDICATION
320

NOF016 DZQL1INX
viii (1) indication verbs
(2) LU_0_TO_3_INDICATION
324

NOF017 DZQL1INX
viii (1) indication verbs
(2) MODE_INDICATION
328

NOF018 DZQL1INX
viii (1) indication verbs
(2) PLU_INDICATION
330

NOF019 DZQL1INX
viii (1) indication verbs
(2) PORT_INDICATION
332
NOF020 DZQL1INX
viii (1) indication verbs
(2) PU_INDICATION
333
NOF021 DZQL1INX
viii (1) indication verbs
(2) REGISTRATION_FAILURE
336
NOF022 DZQL1INX
viii (1) indication verbs
(2) RTP_INDICATION
337
NOF023 DZQL1INX
viii (1) indication verbs
(2) SESSION_INDICATION
341
SEC01 DZQL1INX
viii (1) security verbs
viii, viii, viii, viii
SEC011 DZQL1INX
viii (1) security verbs
(2) DEFINE_LU_LU_PASSWORD
346
SEC012 DZQL1INX
viii (1) security verbs
(2) DEFINE_USERID_PASSWORD
348
SEC013 DZQL1INX
viii (1) security verbs
(2) DELETE_LU_LU_PASSWORD
350
SEC014 DZQL1INX
viii (1) security verbs
(2) DELETE_USERID_PASSWORD
352
PIN01 DZQL1INX
viii (1) APING verbs
viii
PIN011 DZQL1INX
viii (1) APING verbs
(2) APING
356
PIC01 DZQL1INX
viii (1) CPI-C verbs
viii, viii, viii
PIC011 DZQL1INX
viii (1) CPI-C verbs
(2) DEFINE_CPIC_ SIDE_INFO
361
PIC012 DZQL1INX
viii (1) CPI-C verbs
(2) DELETE_CPIC_SIDE_INFO
364
PIC013 DZQL1INX
viii (1) CPI-C verbs
(2) QUERY_CPIC_SIDE_INFO
366
ATM01 DZQL1INX
viii (1) Attach Manager verbs
viii, viii, viii
ATM011 DZQL1INX
viii (1) Attach Manager verbs
(2) DISABLE_ATTACH_MANAGER
370
ATM012 DZQL1INX
viii (1) Attach Manager verbs
(2) ENABLE_ATTACH_MANAGER
371
ATM013 DZQL1INX
viii (1) Attach Manager verbs
(2) QUERY_ATTACH_MANAGER
372
ADJ01 DZQL1INX
viii (1) DEFINE_ADJACENT_LEN_NODE
28
CND01 DZQL1INX
viii (1) DEFINE_CN

				31
COS01	DZQL1INX	viii	(1)	DEFINE_COS
				35
DFS01	DZQL1INX	viii	(1)	DEFINE_DEFAULT_PU
				41
PUD01	DZQL1INX	viii	(1)	DEFINE_DEFAULT_PU
				43
DLD01	DZQL1INX	viii	(1)	DEFINE_DLC
				45
DEF01	DZQL1INX	viii	(1)	DEFINE_DLUR_DEFAULTS
				48
FOC01	DZQL1INX	viii	(1)	DEFINE_FOCAL_POINT
				50
INT01	DZQL1INX	viii	(1)	DEFINE_INTERNAL_PU
				54
LOC01	DZQL1INX	viii	(1)	DEFINE_LOCAL_LU
				57
LSD01	DZQL1INX	viii	(1)	DEFINE_LS
				60
LUD01	DZQL1INX	viii	(1)	DEFINE_LU_0_TO_3
				70
MOD01	DZQL1INX	viii	(1)	DEFINE_MODE
				73
PAR01	DZQL1INX	viii	(1)	DEFINE_PARTNER_LU
				76
POR01	DZQL1INX	viii	(1)	DEFINE_PORT
				79
TPD01	DZQL1INX	viii	(1)	DEFINE_TP
				85
DAD01	DZQL1INX	viii	(1)	DEFINE_ADJACENT_LEN_NODE
				88
CNT01	DZQL1INX	viii	(1)	DELETE_CN
				90
DCO01	DZQL1INX	viii	(1)	DELETE_COS
				92
DDL01	DZQL1INX	viii	(1)	DELETE_DLC
				94
DFP01	DZQL1INX	viii	(1)	DELETE_FOCAL_POINT
				96
DIP01	DZQL1INX	viii	(1)	DELETE_INTERNAL_PU
				98
DLU01	DZQL1INX	viii	(1)	DELETE_LOCAL_LU
				100
DLS01	DZQL1INX	viii	(1)	DELETE_LS
				102
DTU01	DZQL1INX	viii	(1)	DELETE_LU_0_TO_3
				104
DMO01	DZQL1INX	viii	(1)	DELETE_MODE
				106
DPL01	DZQL1INX	viii	(1)	DELETE_PARTNER_LU
				108
DPR01	DZQL1INX	viii	(1)	DELETE_PORT
				110
DTP01	DZQL1INX	viii	(1)	DELETE_TP

112

STD01 DZQL1INX
viii (1) START_DLC
116

STI01 DZQL1INX
viii (1) START_INTERNAL_PU
118, 127

STL01 DZQL1INX
viii (1) START_LS
121

STP01 DZQL1INX
viii (1) START_PORT
123

STC01 DZQL1INX
viii (1) STOP_DLC
125

SIP01 DZQL1INX
viii (1) STOP_INTERNAL_PU

SLS01 DZQL1INX
viii (1) STOP_LS
129

SPO01 DZQL1INX
viii (1) STOP_PORT
131

ACS01 DZQL1INX
viii (1) ACTIVATE_SESSION
133

DCG01 DZQL1INX
viii (1) DEACTIVATE_CONV_GROUP
136

DSS01 DZQL1INX
viii (1) DEACTIVATE_SESSION
138

PAT01 DZQL1INX
viii (1) PATH_SWITCH
141

QCN01 DZQL1INX
viii (1) QUERY_CN
144

QPO01 DZQL1INX
viii (1) QUERY_CN_PORT
149

QCO01 DZQL1INX
viii (1) QUERY_COS
152

QDP01 DZQL1INX
viii (1) QUERY_DEFAULT_PU
156

QDT01 DZQL1INX
viii (1) QUERY_DEFAULTS
158

QDL01 DZQL1INX
viii (1) QUERY_DIRECTORY_LU
160

QDST01 DZQL1INX
viii (1) QUERY_DIRECTORY_STATS
164

QLC01 DZQL1INX
viii (1) QUERY_DLC
166

QLU01 DZQL1INX
viii (1) QUERY_DLUR_LU
171

QDPU01 DZQL1INX
viii (1) QUERY_DLUR_PU
175

QDS01 DZQL1INX
viii (1) QUERY_DLUS
181

QFP01 DZQL1INX
viii (1) QUERY_FOCAL_POINT
186

QLL01 DZQL1INX
viii (1) QUERY_LOCAL_LU
191

QTP01 DZQL1INX
viii (1) QUERY_LOCAL_TOPOLOGY
196

QLS01 DZQL1INX
viii (1) QUERY_LS
201

QUL01 DZQL1INX
viii (1) QUERY_LU_0_TO_3
214
QUM01 DZQL1INX
viii (1) QUERY_MDS_APPLICATION
222
QUS01 DZQL1INX
viii (1) QUERY_MDS_STATISTICS
225
QMD01 DZQL1INX
viii (1) QUERY_MODE
228
QDF01 DZQL1INX
viii (1) QUERY_MODE_DEFINITION
234
QMM01 DZQL1INX
viii (1) QUERY_MODE_TO_COS_MAPPING
238
QNA01 DZQL1INX
viii (1) QUERY_NMVT_APPLICATION
241
QNO01 DZQL1INX
viii (1) QUERY_NODE
244
QNP01 DZQL1INX
viii (1) QUERY_PARTNER_LU
252
QPD01 DZQL1INX
viii (1) QUERY_PARTNER_LU_DEFINITION
258
QPR01 DZQL1INX
viii (1) QUERY_PORT
263
QPU01 DZQL1INX
viii (1) QUERY_PU
271
QRT01 DZQL1INX
viii (1) QUERY_RTP_CONNECTION
276
QSS01 DZQL1INX
viii (1) QUERY_SESSION
283
QST01 DZQL1INX
viii (1) QUERY_STATISTICS
290
QPZ01 DZQL1INX
viii (1) QUERY_TP
292
QPX01 DZQL1INX
viii (1) QUERY_TP_DEFINITION
296
CSL01 DZQL1INX
viii (1) CHANGE_SESSION_LIMIT
302
ISL01 DZQL1INX
viii (1) INITIALIZE_SESSION_LIMIT
306
RSL01 DZQL1INX
viii (1) RESET_SESSION_LIMIT
310
DLI01 DZQL1INX
viii (1) DLC_INDICATION
314
DUI01 DZQL1INX
viii (1) DLUS_INDICATION
315
FCPI01 DZQL1INX
viii (1) FOCAL_POINT_INDICATION
317
LTI01 DZQL1INX
viii (1) LOCAL_TOPOLOGY_INDICATION
319
LSI01 DZQL1INX
viii (1) LS_INDICATION
320
LUO01 DZQL1INX
viii (1) LU_0_TO_3_INDICATION
324
MOI01 DZQL1INX
viii (1) MODE_INDICATION
328

PLI01 DZQL1INX
viii (1) PLU_INDICATION
330
POI01 DZQL1INX
viii (1) PORT_INDICATION
332
PUN01 DZQL1INX
viii (1) PU_INDICATION
333
RFG01 DZQL1INX
viii (1) REGISTRATION_FAILURE
336
RDI01 DZQL1INX
viii (1) RTP_INDICATION
337
SIN01 DZQL1INX
viii (1) SESSION_INDICATION
341
DRO01 DZQL1INX
viii (1) DEFINE_LU_LU_PASSWORD
346
USP01 DZQL1INX
viii (1) DEFINE_USERID_PASSWORD
348
LUL01 DZQL1INX
viii (1) DELETE_LU_LU_PASSWORD
350
URP01 DZQL1INX
viii (1) DELETE_USERID_PASSWORD
352
APG01 DZQL1INX
viii (1) APING
356
CPS01 DZQL1INX
viii (1) DEFINE_CPIC_SIDE_INFO
361
CPD01 DZQL1INX
viii (1) DELETE_CPIC_SIDE_INFO
364
QCS01 DZQL1INX
viii (1) QUERY_CPIC_SIDE_INFO
366
ALI01 DZQL1INX
viii (1) ALERT_INDICATION
398
FPI01 DZQL1INX
viii (1) FP_INDICATION
399
MMU01 DZQL1INX
viii (1) MDS_MU_RECEIVED
393
NMV01 DZQL1INX
viii (1) NMVT_RECEIVED
400
SMU01 DZQL1INX
viii (1) SEND_MDS_MU
395
TMS01 DZQL1INX
viii (1) TRANSFER_MS_DATA
390
DAM01 DZQL1INX
viii (1) DISABLE_ATTACH_MANAGER
370
EAM01 DZQL1INX
viii (1) ENABLE_ATTACH_MANAGER
371
QAM01 DZQL1INX
viii (1) QUERY_ATTACH_MANAGER
372
ALUNSOC DZQL1INX
viii (1) alerts, unsolicited
390
CHILD DZQL1INX
viii (1) children
28
COMPILE DZQL1INX
viii (1) compiling and linking a Node Operator Facility program
6, 376
CONNET DZQL1INX
viii (1) Connection Network
15, 144

DATALST DZQL1INX
viii (1) **data_lost** indicator
14
DETINF DZQL1INX
viii (1) detailed information
13
DBCS DZQL1INX
viii (1) double-byte character set support
x
DYNALNK DZQL1INX
viii (1) dynamically linking to a DLL
376
DLL DZQL1INX
viii (1) DLL (dynamic link library)
viii, 376, 382
DLLA DZQL1INX
viii (1) dynamic link library (DLL)
ENTPTS DZQL1INX
viii (1) entry points
viii, viii, viii, viii
ENPINTR DZQL1INX
viii (1) entry points
(2) introduction
5, 375
ENPIND DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
viii, viii, viii, viii, viii, viii, viii, viii, viii
ENPWNF DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinNOF()**
18
ENPASY DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinAsyncNOF()**
19
ENPASYX DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinAsyncNOFEx()**
20
ENPCANC DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinNOFCancelAsyncRequest()**
21
EPNFCLN DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinNOFCleanup()**
22
EPNFSTP DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinNOFStartup()**
23
ENPREG DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinNOFRegisterIndicationSink()**
14, 24
ENPUNRG DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinNOFUnregisterIndicationSink()**
14, 25
ENPGET DZQL1INX
viii (1) entry points
(2) for Node Operator Facility verbs
(3) **WinNOFGetIndication()**
14, 26, 387
ENPMS DZQL1INX
viii (1) entry points
(2) for Management Services verbs
viii, viii, viii, viii, viii
ENPWMS DZQL1INX
viii (1) entry points
(2) for Management Services verbs
(3) **WinMS()**

380
EPCLNUP DZQL1INX
viii (1) entry points
(2) for Management Services verbs
(3) **WinMSCleanup()**
381
EPSTUP DZQL1INX
viii (1) entry points
(2) for Management Services verbs
(3) **WinMSStartup()**
382
EPRGAPP DZQL1INX
viii (1) entry points
(2) for Management Services verbs
(3) **WinMSRegisterApplication()**
383
EPUGAPP DZQL1INX
viii (1) entry points
(2) for Management Services verbs
(3) **WinMSUnregisterApplication()**
385
ENTLINK DZQL1INX
viii (1) entry points
(2) linkage information
377
FP DZQL1INX
viii (1) focal point
viii, viii, viii, viii, viii
EXPFP DZQL1INX
viii (1) focal point
(2) explicit
50
IMPPRFP DZQL1INX
viii (1) focal point
(2) implicit primary
50
IMPBUFP DZQL1INX
viii (1) focal point
(2) implicit backup
50
DOMFP DZQL1INX
viii (1) focal point
(2) domain
50
HOSTFP DZQL1INX
viii (1) focal point
(2) host
50
GPF DZQL1INX
viii (1) general protection fault
6, 376
HILITE DZQL1INX
viii (1) highlighting, how used in this book
x
HPR DZQL1INX
viii (1) HPR (high-performance routing)
141
LENNODE DZQL1INX
viii (1) LEN node
15
LIMRES DZQL1INX
viii (1) limited resource
65
LDT DZQL1INX
viii (1) local descriptor table
6, 375
LUPOOL DZQL1INX
viii (1) LU pool
70
NODEROW DZQL1INX
viii (1) node row (in a class of service definition)
35
PSEUDO DZQL1INX
viii (1) pseudonym files
viii, viii
PSWINMS DZQL1INX
viii (1) pseudonym files
(2) WINMS.H
377
PSWINNF DZQL1INX
viii (1) pseudonym files

(2) WINNOF.H
7

SATF DZQL1INX
viii (1) SATF (shared-access transport facility)
15

SUMINF DZQL1INX
viii (1) summary information
13

TGROW DZQL1INX
viii (1) TG row (in a class of service definition)
35

UNSOCAL DZQL1INX
viii (1) unsolicited alerts
390

VCB DZQL1INX
viii (1) verb control block
viii, viii

VCBCOM DZQL1INX
viii (1) verb control block
(2) common fields
10

VCBINT DZQL1INX
viii (1) verb control block
(2) introduction
5, 6, 375

WILD DZQL1INX
viii (1) wildcard entries
28, 88

WINMSH DZQL1INX
viii (1) WINMS.H file
377

WINNOFH DZQL1INX
viii (1) WINNOF.H file
7

WINLIB DZQL1INX
viii (1) WINNOF32.LIB file
6, 376

WINNOF DZQL1INX
viii (1) **WinNOF()**
18

WINANOF DZQL1INX
viii (1) **WinAsyncNOF()**
19

WINANFS DZQL1INX
viii (1) **WinAsyncNOFEx()**
20

WINCANC DZQL1INX
viii (1) **WinNOFCancelAsyncRequest()**
21

WFCLNUP DZQL1INX
viii (1) **WinNOFCleanup()**
22

WFSTUP DZQL1INX
viii (1) **WinNOFStartup()**
23

WINREG DZQL1INX
viii (1) **WinNOFRegisterIndicationSink()**
14, 24

WINUNRG DZQL1INX
viii (1) **WinNOFUnregisterIndicationSink()**
14, 25

WINGET DZQL1INX
viii (1) **WinNOFGetIndication()**
14, 26, 387

WINMS DZQL1INX
viii (1) **WinMS()**
380

CLEANUP DZQL1INX
viii (1) **WinMSCleanup()**
381

STARTUP DZQL1INX
viii (1) **WinMSStartup()**
382

REGAPP DZQL1INX
viii (1) **WinMSRegisterApplication()**
383

UNRGAPP DZQL1INX
viii (1) **WinMSUnregisterApplication()**
385

XID DZQL1INX
viii (1) XID

			15
XID3	DZQL1INX		
		viii	(1) XID3
			15

Tables

id	File	Page	References
PORTYPE	DZQL1NCF	46	1

Processing Options

Runtime values:

Document fileid	DZQL1MST SCRIPT
Document type	USERDOC
Document style	IBMXAGD
Profile	EDFPRF40
Service Level	0030
SCRIPT/VS Release	4.0.0
Date	96.03.25
Time	15:39:54
Device	PSA
Number of Passes	3
Index	YES
SYSVAR D	YES
SYSVAR V	IDD2
SYSVAR X	YES
SYSVAR 1	ISO9000

Formatting values used:

Annotation	NO
Cross reference listing	YES
Cross reference head prefix only	NO
Dialog	LABEL
Duplex	YES
DVCF conditions file	(none)
DVCF value 1	ISO9000
DVCF value 2	(none)
DVCF value 3	(none)
DVCF value 4	(none)
DVCF value 5	(none)
DVCF value 6	(none)
DVCF value 7	(none)
DVCF value 8	(none)
DVCF value 9	(none)
Explode	NO
Figure list on new page	YES
Figure/table number separation	YES
Folio-by-chapter	NO
Head 0 body text	Part
Head 1 body text	Chapter
Head 1 appendix text	Appendix
Hyphenation	NO
Justification	NO
Language	ENGL
Keyboard	395
Layout	OFF
Leader dots	YES
Master index	(none)
Partial TOC (maximum level)	3
Partial TOC (new page after)	RHPAGE
Print example id's	NO
Print cross reference page numbers	YES
Process value	(none)
Punctuation move characters	.,
Read cross-reference file	(none)
Running heading/footing rule	NONE
Show index entries	NO
Table of Contents (maximum level)	3
Table list on new page	YES
Title page (draft) alignment	RIGHT
Write cross-reference file	(none)

Imbed Trace

Page 0	DZQL1SET
Page i	DZQL1EDN
Page vi	DZQL1NOT
Page viii	CMTRADEN
Page viii	DZQL1INX
Page viii	DZQL1PRE
Page 4	DZQL1CH1
Page 7	DZQL1CH2
Page 16	DZQL1ENP
Page 26	DZQL1NCF
Page 113	DZQL1ACT
Page 142	DZQL1QRY
Page 300	DZQL1SLM
Page 312	DZQL1IND
Page 343	DZQL1SEC
Page 353	DZQL1CPI
Page 368	DZQL1ATM
Page 374	DZQL1MSI
Page 377	DZQL1MSE
Page 387	DZQL1MGT
Page 402	DZQL1APA
Page 404	DZQL1GLO
Page 447	RCFADDR
Page 449	RCFADDR
Page 452	DZQL1C2