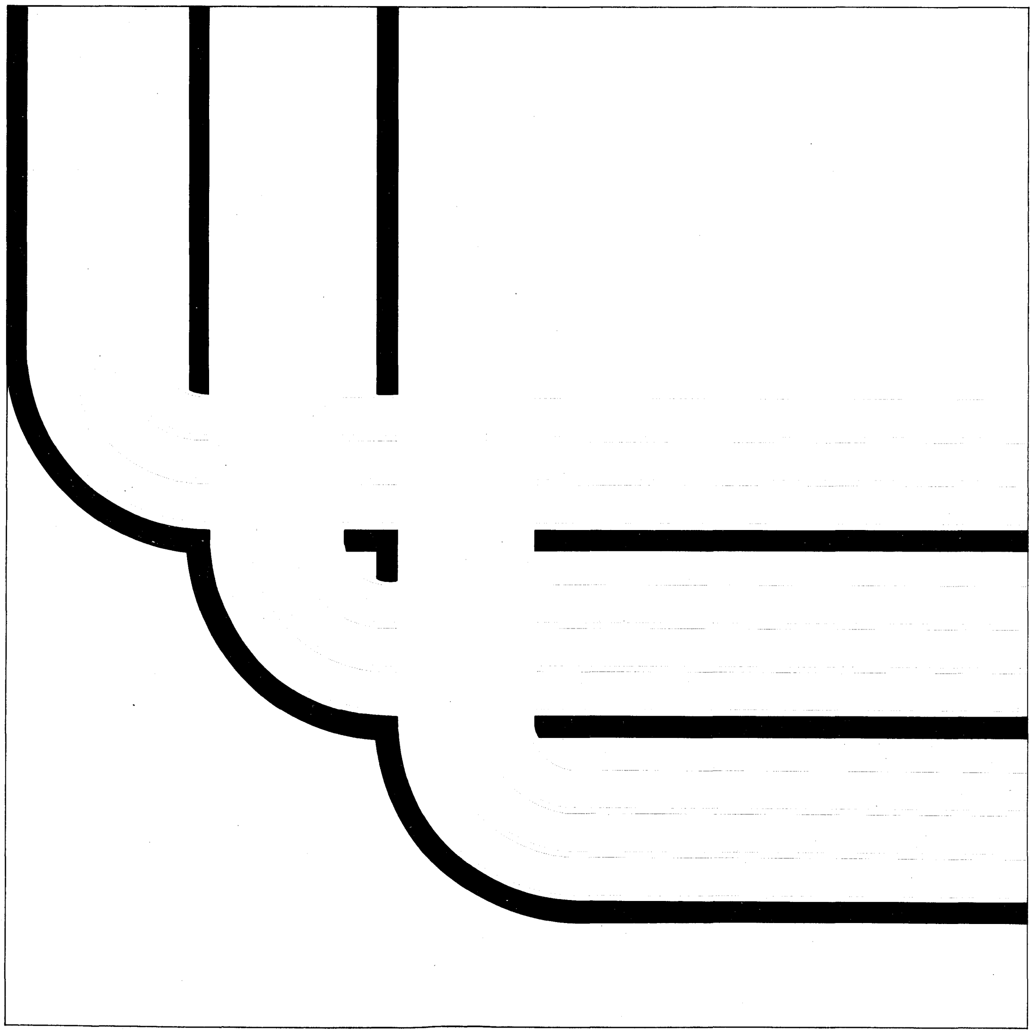
Application System/400™

SC41-0027-00

System Programmer's Communications Interface Guide

Version 2



System and Application Support

Take Note!

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

First Edition (May 1991)

This edition applies to the licensed program IBM Operating System/400 (Program 5738-SS1), Version 2 Release 1 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the proper edition for the level of the product.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, you may address your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

Contents

Notices . . . vii
Programming Interfaces . . . vii

About This Guide . . . ix
Who Should Use This Guide . . . ix

Part 1. User-Defined Communications

Chapter 1. Introduction to User-Defined Communications . . . 1-1
Overview . . . 1-1
  User-Defined Communications Callable Routines . . . 1-2
  Input/Output Buffers and Descriptors . . . 1-2
  Data Queue . . . 1-2
Important Concepts . . . 1-3
Relationship to Communications Standards . . . 1-4
LAN Considerations . . . 1-6
X.25 Considerations . . . 1-7

Chapter 2. User-Defined Communications Support APIs . . . 2-1
QOLELINK . . . 2-3
  Parameter List . . . 2-3
  Description of Function . . . 2-5
  Return and Reason Codes . . . 2-6
QOLDLINK . . . 2-8
  Parameter List . . . 2-8
  Description of Function . . . 2-8
  Return and Reason Codes . . . 2-9
QOLSETF . . . 2-10
  Parameter List . . . 2-10
  Description of Function . . . 2-10
  General Filter Rules . . . 2-15
  Return and Reason Codes . . . 2-15
QOLSEND . . . 2-17
  Parameter List . . . 2-17
  Description of Function . . . 2-20
  LAN Output Operations . . . 2-20
  X.25 SVC and PVC Output Operations . . . 2-22
  Return and Reason Codes . . . 2-35
QOLRECV . . . 2-40
  Parameter List . . . 2-40
  Description of Function . . . 2-44
  LAN Input Operations . . . 2-44
  X.25 SVC and PVC Input Operations . . . 2-46
  Return and Reason Codes . . . 2-52
QOLQLIND . . . 2-57
  Parameter List . . . 2-57
  Description of Function . . . 2-57
  Return and Reason Codes . . . 2-60
QOLTIMER . . . 2-62
  Parameter List . . . 2-62
  Description of Function . . . 2-63

Return and Reason Codes 2-64

Chapter 3. Programming Design Considerations 3-1
Jobs 3-1
Application Program Feedback 3-5
Synchronous and Asynchronous Operations 3-5
Programming Languages 3-5
Starting and Ending Communications 3-6
Programming Considerations for X.25 Applications 3-6
X.25 Packet Types Supported 3-6
Connections 3-8
Switched Virtual Circuit (SVC) Connectivity 3-10
Permanent Virtual Circuit (PVC) Connectivity 3-11
Sending and Receiving Data Packets 3-12
Using Connection Identifiers 3-14
AS/400 System X.25 Call Control 3-29
Performance Considerations 3-31
Programming Considerations for LAN Applications 3-32
LAN Frames Supported 3-32
Configuration 3-33
Inbound Routing Information 3-34
End-to-End Connectivity 3-34
Sending and Receiving Data 3-34
Ethernet to Token-Ring Conversion and Routing 3-35
Performance Considerations 3-35
Data Queue Considerations 3-36
User Space Considerations 3-38

Chapter 4. Application Programming Examples 4-1
X.25 Overview 4-1
User-Defined Communications Support Overview 4-1
C/400 Compiler Listings 4-3

Chapter 5. Application Debugging 5-1
System Services and Tools 5-1
Program Debug 5-1
Work with Communications Status 5-1
Display Job Log 5-1
Display Connection Status 5-1
Display Inbound Routing Data 5-2
Work with Communications Trace 5-2
Work with Error Log 5-2
Dump System Object to View User Spaces 5-3
Error Codes 5-11
LAN Error Codes 5-11
X.25 Error Codes 5-12
Common Errors 5-15

Chapter 6. Configuration and Additional Information 6-1
Configuring User-Defined Communications Support 6-1
Link 6-1
Data Queue 6-2
Data Queue Entries 6-2
General Format 6-2
Enable-Complete Entry 6-3

Disable-Complete Entry 6-3
Permanent-Link-Failure Entry 6-4
Incoming-Data Entry 6-4
Timer-Expired Entry 6-5

Part 2. Virtual Terminal Application Programming Interfaces

Chapter 7. Introduction to Virtual Terminal APIs 7-1
Implementation of Distributed 5250 Emulation Model 7-1

Chapter 8. Getting ready for using the VT APIs 8-1
Step 1 - Setting the Number of Automatically Created Virtual Terminals . . 8-1
Security Considerations 8-2
Step 2 - Setting the Limit Security Officer (QLMTSECOFR) Value 8-2
Step 3 - Creating User Profiles 8-3
Creating Your Own Virtual Controllers and Devices 8-3
Developing Server and Client Programs 8-4

Chapter 9. Before Using VT APIs 9-1
Work Station Types 9-1
Virtual Terminal Data 9-1
Data Queues 9-1
AS/400 Job Information 9-2
AS/400 Subsystem Information 9-2

Chapter 10. Virtual Terminal APIs 10-1
Open Virtual Terminal Path API (QTVOPNVT) 10-1
Supported Work Station Types and Models 10-3
Read from Virtual Terminal API (QTVRDVT) 10-4
Read Operation Codes 10-6
Write to Virtual Terminal API (QTVWRTVT) 10-8
Write Operation Codes 10-9
Send Request for OS/400 Function API (QTVSNDRQ) 10-10
Close Virtual Terminal Path API (QTVCLOVT) 10-11
VT API Error Reporting 10-11
Error Code Parameter 10-11

Chapter 11. VT APIs Run-Time Example 11-1

Part 3. Appendixes

Appendix A. Additional APIs A-1

Appendix B. 5250 Data Stream and Keystroking Enhancements B-1
Changes to the 5250 Functions Reference Manual B-1
Enhancements to 5250 Data Stream Commands and Orders B-1
New 5250 Data Stream Commands and Orders B-2
New 5250 Keystroke Processing B-7

Bibliography H-1
AS/400 Manuals H-1
Communications Related Manuals H-2

Index X-1

Notices

Application System/400	AS/400	C/400
COBOL/400	IBM	Operating System/400
OS/2	OS/400	PS/2
RPG/400	400	

Programming Interfaces

This *System Programmer's Communications Interface Guide* is intended for system programmers who will be doing specialized communications programming on the AS/400 system. It primarily contains general-use programming interfaces, which allow the customer to write programs that use the services of the OS/400 (Program 5738-SS1), Version 2 Release 1 Modification 0.

About This Guide

This guide describes the application program interfaces (APIs) for user-defined communications and other communications-oriented APIs on the AS/400 system.

Part 1 contains the information needed to write user-defined communications applications, programming examples, and debugging information.

Part 2 contains the information needed to use the Virtual Terminal (VT) APIs, which allows an AS/400 application program to interact with an AS/400 system that is performing work station input/output (I/O).

Who Should Use This Guide

This guide is intended for system programmers who will be doing specialized communications programming on the AS/400 system. Those who configure user-defined communications and programmers who write to the user-defined communications or VT APIs should use this guide.

You should be familiar with general communications concepts. AS/400 communications concepts are covered in the *System Concepts*, GC41-9802, manual. In addition, specific communications topics are discussed in the online index search. For more information on basic communications, you can also refer to the Discover/ IBM AS/400 course in the communications module. The Discover/ IBM AS/400 course may be ordered separately. If you are interested in IBM customer education, contact your IBM marketing representative regarding customer education classes held at IBM education centers.

You should also know how to use the AS/400 menus and commands and how to write applications on the AS/400 system.

You should be familiar with CCITT X.25, IEEE 802.2, IEEE 802.3, IEEE 802.5, or Ethernet Version 2 protocols.

Refer to the “Bibliography” for a list of manuals related to the information in this guide.

Part 1. User-Defined Communications

Chapter 1. Introduction to User-Defined Communications

User-defined communications is a set of application program interfaces (APIs) that are part of Operating System/400* (OS/400*). These callable programs allow customers to write their own communications protocol stacks above the AS/400* data link and physical layer support. The term, user-defined communications, will be used to describe this new communications protocol support.

This publication defines user-defined communications, and describes how to write protocols using the APIs. In addition, Chapter 4, "Application Programming Examples" describes two C language program examples that illustrate the use of the various APIs while performing a simple file transfer between two systems attached to an X.25 packet switched network.

Overview

The user-defined communications APIs are callable routines that allow an application program to send and receive data, and do specialized functions such as setting timers.

Figure 1-1 shows an overview of the user-defined communications support.

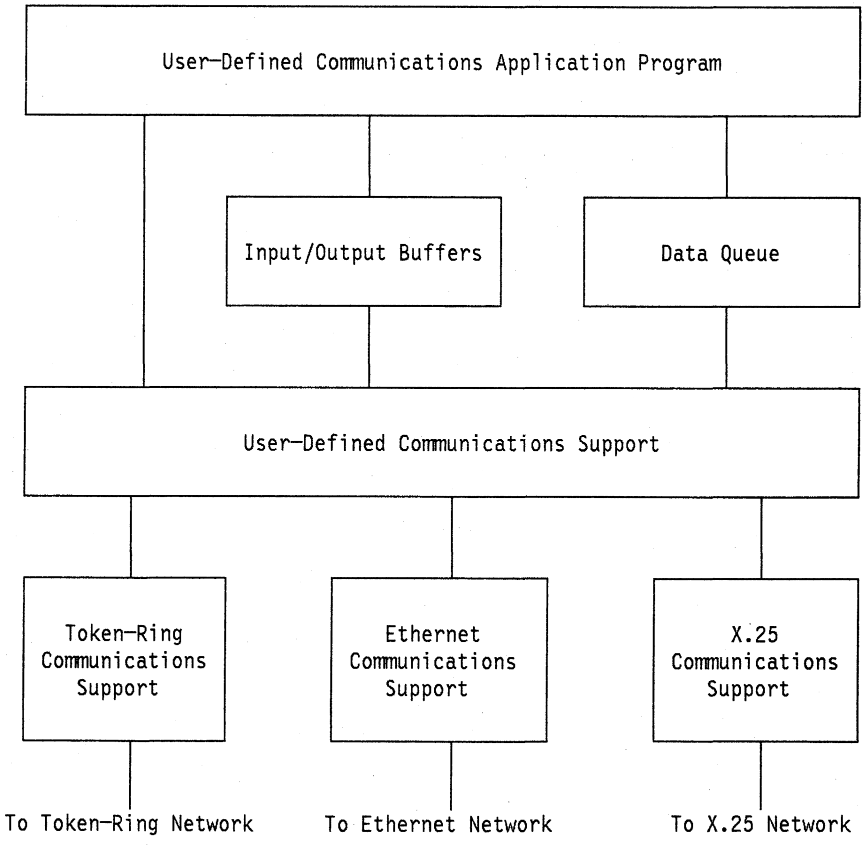

User-Defined Communications Application Program
Input/Output Buffers
Data Queue
User-Defined Communications Support
Token-Ring Communications Support
Ethernet Communications Support
X.25 Communications Support
To Token-Ring Network
To Ethernet Network
To X.25 Network

Figure 1-1. User-Defined Communications Support

As Figure 1-1 on page 1-1 indicates, a user-defined communications application program needs to coexist with the following:

- user-defined communications support
- input/output buffers and descriptors
- a data queue

User-Defined Communications Callable Routines

Callable routines are provided that allow a user-defined communications application program to start, perform, and end communications, and perform specialized functions such as setting timers. These routines are listed below and are discussed in detail in Chapter 2, "User-Defined Communications Support APIs."

- QOLELINK — start communications
- QOLDLINK — end communications
- QOLSETF — provide routing information for inbound data
- QOLSEND — send data
- QOLRECV — receive data
- QOLQLIND — retrieve line description information
- QOLTIMER — set or cancel a timer

Input/Output Buffers and Descriptors

The input/output buffers and descriptors are user space objects (*USRSPC) that contain and describe the data a user-defined communications application program is sending or receiving. There are separate buffers and descriptors for input and output.

When a user-defined communications application program wishes to send data, it fills the output buffer with data and provides a description of that data in the output buffer descriptor. Similarly, when a user-defined communications application program receives data, the user-defined communications support fills the input buffer with data and provides a description of that data in the input buffer descriptor.

Callable routines are provided that allow a user-defined communications application program to manipulate the data in the user spaces. Some of these routines are listed below.

- QUSPTRUS — get a pointer to the user space
- QUSCHGUS — change contents of the user space
- QUSRTVUS — retrieve contents of the user space

See the *System Programmer's Interface Reference* for more information on the user space APIs.

Data Queue

The data queue is used by the user-defined communications support to inform a user-defined communications application program of some action to take or of an activity that has been completed.

Callable routines are provided that allow a user-defined communications application program to manipulate the data queue. Some of these routines are listed below.

- QSNDDTAQ — send an entry to the data queue
- QRCVDTAQ — receive an entry from the data queue

- QCLRDTAQ — clear all entries from the data queue

See the *CL Programmer's Guide* for more information on the data queue APIs.

Important Concepts

Listed below are concepts that are important in understanding the information contained in this guide.

Link: The logical path between a user-defined communications application program and a communications line. A link is made up of the following communications objects:

- an X.25, token-ring, or Ethernet line description
- a network controller description
- a network device description of type *USRDFN

Communications handle: The name a user-defined communications application program assigns and uses to refer to a link.

Enable: The process of setting up and activating a link for input and output on a communications line.

Disable: The process of deactivating a link so input and output is no longer possible on a communications line.

Filter: The technique used to route inbound data to a link that is enabled by a user-defined communications application program.

Connection: The logical communication path from one computer system to another. For example, a switched virtual circuit (SVC) connection on an X.25 network.

Connection identifier: A local identifier (ID) that a computer system uses to distinguish one connection from another. When using the user-defined communications support on the AS/400 system, a connection ID is made up of a user connection end point ID and a provider connection end point ID.

User connection end point ID (UCEP ID): The portion of the connection ID that the user-defined communications application program uses to identify the connection. For example, data received by the user-defined communications application program will be on the UCEP ID portion of the connection ID.

Provider connection end point ID (PCEP ID): The portion of the connection ID that the user-defined communications support uses to identify the connection. For example, data sent by the user-defined communications application program will be on the PCEP ID portion of the connection ID.

Connection-oriented service: A method of operation where a connection to the remote computer system must first be established before data can be sent to it or received from it. User-defined communications support provides connection-oriented service over X.25 networks only.

Connectionless service: A method of operation where data can be sent to and received from the remote computer system without establishing a connection to it. User-defined communications support provides connectionless service over token-ring and Ethernet networks only. To LANs, connectionless service is also known as unacknowledged service.

Relationship to Communications Standards

Figure 1-2 shows the structure of advanced program-to-program communications (APPC) on the AS/400 system and its relationship to the International Standards Organization (ISO) protocol model. Note that only the application layer above the APPC protocol code is available for definition. The APPC functional equivalents of the ISO presentation, session, networking, transport, data link, and physical layers are performed by OS/400 or licensed internal code, and you may not replace or change them. Contrast this with Figure 1-3 on page 1-5 which shows how much more of the protocol is defined by the user-defined communications application than by the APPC application.

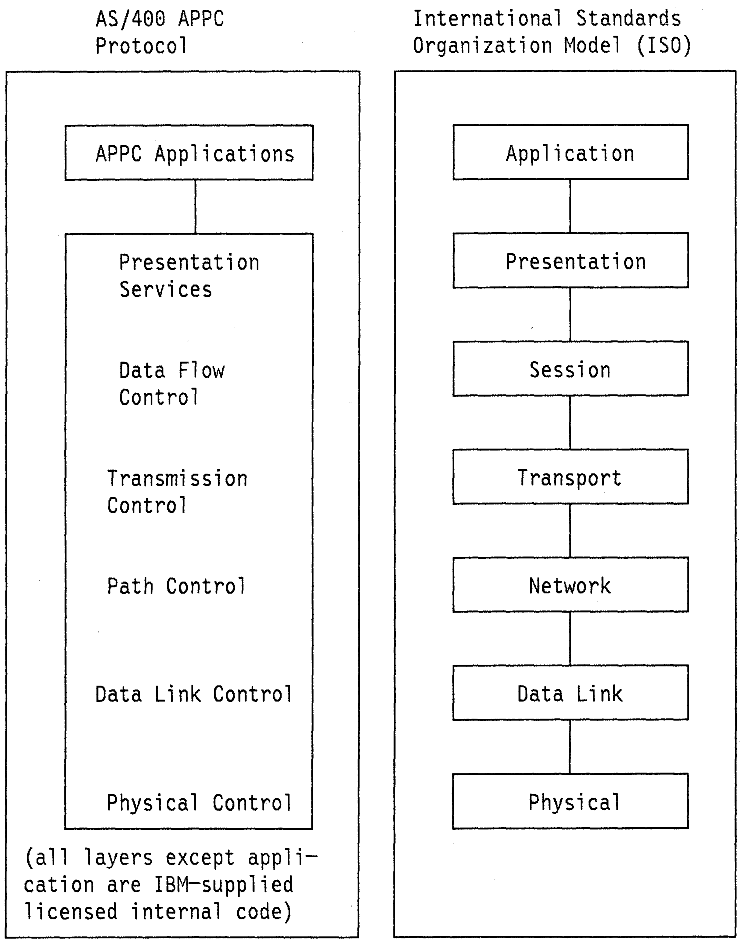


Figure 1-2. AS/400 APPC versus ISO Model

Figure 1-3 on page 1-5 shows the new structure for user-defined communications and its relationship to the International Standards Organization (ISO) protocol model. Note that the available AS/400 data links and physical layers limit user-defined communications to run over LAN (token-ring or Ethernet) or X.25 links, but the portion of the protocol above the data link layer is completely open to the user-defined communications application. In addition, these same X.25 and LAN links may be shared between the user-defined communications application and other AS/400 communications protocols that support X.25 and LAN lines.

Examples include Systems Network Architecture (SNA), asynchronous communications, Transmission Control Protocol/Internet Protocol (TCP/IP), and Open Systems Interconnection (OSI).

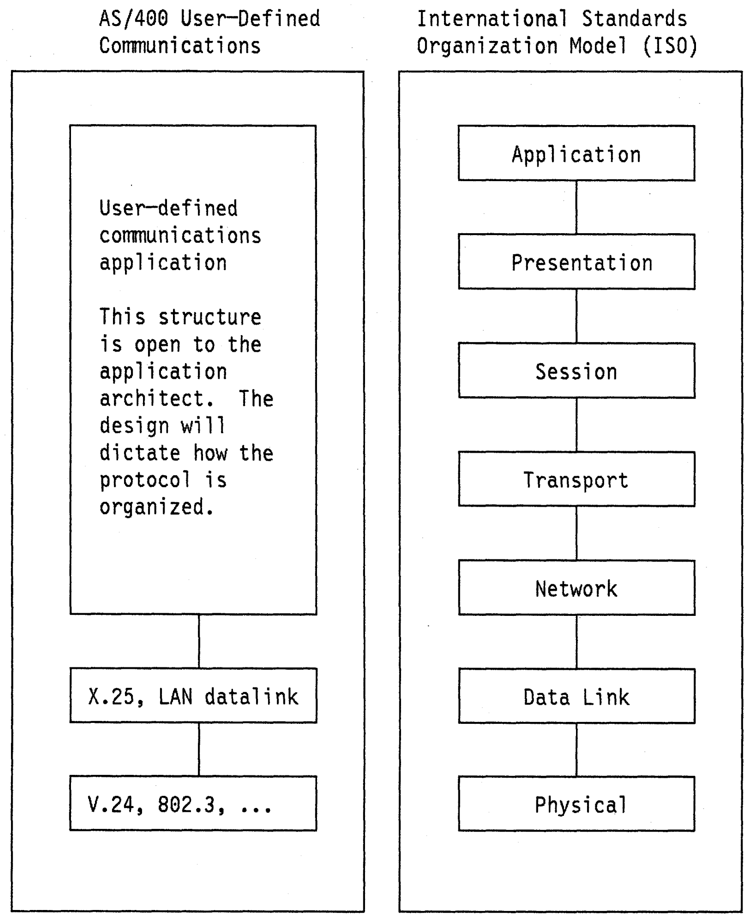

AS/400 User-Defined Communications
International Standards Organization Model (ISO)
User-defined communications application
This structure is open to the application architect. The design will dictate how the protocol is organized.
X.25, LAN datalink
V.24, 802.3, ...
Application
Presentation
Session
Transport
Network
Data Link
Physical

Figure 1-3. AS/400 User-defined versus ISO Model

Protocols that run over local area networks or X.25 networks may be written completely in high-level languages such as C/400*, COBOL/400*, or RPG/400*. Protocols that are currently running on other systems may be written to run on the AS/400 system. It is now possible for a customer or an IBM* Business Partner to write both non-SNA LAN or X.25 packet layer protocols on the AS/400 system.

Configuration instructions also need to be supplied with the user-defined communications application. User-defined communications support simply opens a *pathway* to the system data links. It is up to the protocol developer to supply any configuration instructions that are in addition to the data link or physical layer definition. Data link and physical layer definitions are defined when the user performs a Create Line Description (token-ring) (CRTLINTRN), Create Line Description (Ethernet) (CRTLINETH), or Create Line Description (X.25) (CRTLINX25).

Table 1-1 on page 1-6 outlines the difference between standard AS/400 communications configuration and user-defined communications configuration, such as the AS/400 APPC protocol.

Table 1-1. User-defined Communications Configuration

Object	APPC Communications	User-Defined Communications
Line Description	SDLC, LAN, X.25 lines. Contains local port information for AS/400 communication IOP (hardware address, maximum frame size, exchange identifier (XID), local recovery information, ...).	LAN, X.25 lines. Same as APPC except some of the information does not apply to user-defined communications.
Controller Description	APPC, host controllers. Describes remote system and parameters must match the remote hardware (hardware address, XID, ...).	Network controller. Pathway into network. Only one specific parameter - X.25 time-out value.
Device Description	APPC device. Describes remote logical unit (LU), and parameters must match partner LU (remote location name, local location name, ...).	Network device. Only describes the communications method or type (for example, TCP/IP, OSI, or user-defined communications).
Mode Description and Class-of-Service (COS)	Required.	Not available

This table shows that although an APPC network would require one APPC controller description to describe each remote system in the network, user-defined communications would only require one network controller for communications with an entire network of remote systems. This allows user-defined communications application writers to easily manage their networks. However, system-specific configuration information must be part of the user-defined communication application itself and is not supplied by IBM.

Although the configuration objects are different from the standard objects for AS/400 communications protocols, LAN and X.25 lines may be shared between user-defined communications and any other protocols that support those same line types. For example, APPC may run over a token-ring line and use the '04'X Service Access Point (SAP). TCP/IP might run at the same time using the 'AA'X SAP. A user-defined communications application might be written to use the '22'X SAP, and run at the same time as the first two. All three protocols may be active at the same time across the same physical media.

LAN Considerations

Three LAN types are supported: token-ring (IEEE 802.5), Ethernet (IEEE 802.3), and Ethernet Version 2.

The user-defined communications application has access to Class I, Type 1 unnumbered information (UI) frames through the user-defined communications application. This *connectionless* service is commonly referred to as *datagram* support where protocol data units are exchanged between end points without establishing a data link connection first.

The Class I, Type 1 operations, Test and XID frames, are not supported in user-defined communications. Any XID or Test frames received by the AS/400 physical layer are processed by the IOP and never reach the user-defined communications application.

LAN frames are routed by filtering the incoming data based on the inbound routing data that is defined by the user-defined communications application. The filters are hierarchical and are set up by the user-defined communications application before communications is started.

The possible settings for LAN inbound routing data (filters) are shown below from least selective to most selective.

- Destination Service Access Point (DSAP)
- DSAP, Source Service Access Point (SSAP), and optional frame type
- DSAP, SSAP, optional frame type, and adapter address

Because user-defined communications does not allow applications to define the data link and physical layers, the entire token-ring or Ethernet frame is not available to applications. The following fields are the parts of the LAN frame that are available to the user-defined communications support:

- DSAP
- SSAP
- Destination Address (DA)
- Routing Information field (RI)
- Priority control
- Access control
- Data

X.25 Considerations

X.25 user-defined communications support includes access to both permanent virtual circuits (PVCs) and switched virtual circuits (SVCs).

Over X.25 networks, the user-defined communications application may initiate and accept X.25 calls, send and receive data, and reset and clear connections. For additional information on X.25 support, refer to "Programming Considerations for X.25 Applications" on page 3-6.

X.25 packets are routed by filtering the incoming call request based on the inbound routing data that is defined by the user-defined communications application. The filters are hierarchical and are set up by the user-defined communications application before communications is started.

The possible settings for X.25 inbound routing data (filters) are shown below from least selective to most selective.

- Protocol identifier (PID)
- PID, and calling Data Terminal Equipment (DTE) address

Chapter 2. User-Defined Communications Support APIs

User-defined communications support is made up of seven callable application program interfaces (APIs) that provide services for the user-defined communications application program.

All parameters must be passed to each API by reference (for example, by an address to the information). An input parameter is used to pass information to an API. An output parameter is used to receive information from an API.

When control returns from an API, the status of the operation will be located in the return code and reason code output parameters. Return codes are 4-byte numbers that determine the recovery action to take. They are grouped into the following categories:

- 00 — operation successful, no recovery needed
- 80 — nonrecoverable error, need to disable link
- 81 — nonrecoverable error, do not need to disable link
- 82 — recoverable error, enable link failed
- 83 — recoverable error, see recovery actions

Reason codes are 4-byte numbers that determine what error occurred. They are grouped into the following categories:

- 0000 — no error
- 1xxx — parameter validation error
- 20xx — line, controller, or device description error
- 22xx — data queue error
- 24xx — buffer or buffer descriptor error
- 30xx — link state error
- 32xx — connection state error
- 34xx — timer state error
- 4xxx — communication error
- 8xxx — application error
- 9999 — a condition in which an Authorized Program Analysis Report (APAR) may be submitted

Note: 'x' represents any decimal number. For example, 1xxx represents the range 1000 - 1999.

In addition to parameters, some of the APIs also use input/output buffers and descriptors to receive information from, and pass information to, the user-defined communications application program. For example, when a user-defined communications application program sends data, it fills the output buffer with data and provides a description of that data in the output buffer descriptor. Similarly, when a user-defined communications application program receives data, the user-defined communications support fills the input buffer with data and provides a description of that data in the input buffer descriptor.

The input/output buffers and descriptors are user space objects (*USRSPC) that are created by user-defined communications support when a link is enabled and deleted by user-defined communications support when that link is disabled. For each link, there is an input buffer, input buffer descriptor, output buffer, and output buffer descriptor. The structure of the input buffer and output buffer are the same. Likewise, the structure of the input buffer descriptor and output buffer descriptor are the same. Figure 2-1 shows the structure of a buffer and the corresponding buffer descriptor.

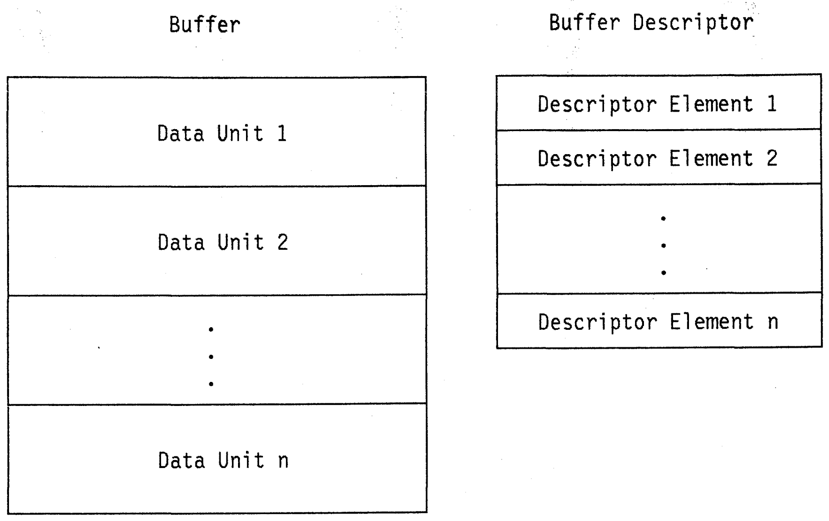


Figure 2-1. Structure of Buffer and Buffer Descriptor

The buffer is divided into equally sized, contiguous sections called data units (DUs). A DU in the output buffer contains data to be sent on the network. A DU in the input buffer contains data received off the network. The size of each DU, as well as the number of DUs created, will be returned from the QOLELINK program when the link is enabled.

The buffer descriptor is divided into equally sized, contiguous sections called descriptor elements. Each descriptor element describes the data in the corresponding DU of the buffer. For example, descriptor element 1 describes the data in data unit 1 of the buffer. The size of each descriptor element is 32 bytes.

Uses of the input/output buffers and descriptors, as well as the formats of the DUs and descriptor elements, are discussed later in this chapter.

QOLELINK

```
CALL QOLELINK(return code,
              reason code,
              data unit size,
              data units created,
              LAN user data size,
              X.25 user data size,
              input buffer,
              input buffer descriptor,
              output buffer,
              output buffer descriptor,
              key length,
              key value,
              queue name,
              line description,
              communications handle)
```

Parameter List

Table 2-1 (Page 1 of 3). QOLELINK Parameter List

Parameter	Use	Type	Description
return code	output	BINARY(4)	Specifies the recovery action to take. See "Return and Reason Codes" on page 2-6.
reason code	output	BINARY(4)	Specifies the error that occurred. See "Return and Reason Codes" on page 2-6.
data unit size	output	BINARY(4)	Specifies the total number of bytes allocated for each data unit in the input and output buffers. For token-ring links, this includes user data (LAN user data size parameter), logical link control (LLC) information, and optional routing information. For Ethernet links, this includes user data (LAN user data size parameter) and LLC information. For X.25 links, this includes user data (X.25 user data size parameter).
data units created	output	BINARY(4)	Specifies the number of data units created for the input buffer and the output buffer. This parameter also specifies the number of elements created for the input buffer descriptor and the output buffer descriptor. Currently, 8 is the only possible value for this parameter. However, the user-defined communications application program should be written to avoid having to recompile should this value ever change.
LAN user data size	output	BINARY(4)	Specifies the number of bytes allocated for token-ring or Ethernet user data in each data unit of the input and output buffers, not including logical link control (LLC) information and optional routing information. The content of this parameter is only valid when enabling a token-ring or Ethernet link. **Note:** The maximum amount of token-ring or Ethernet user data that can be sent or received in each data unit is determined on a service access point basis in the line description and by the 1502 byte maximum for Ethernet Version 2 frames, and may be less than LAN user data size. See "QOLQLIND" on page 2-57 for more information.

Table 2-1 (Page 2 of 3). QOLELINK Parameter List

Parameter	Use	Type	Description
X.25 user data size	input	BINARY(4)	Specifies the number of bytes allocated for X.25 user data in each data unit of the input and output buffers. This is equal to the maximum amount of X.25 user data that can be sent or received in each data unit. Any value between 512 and 4096 may be used. The content of this parameter is only valid when enabling an X.25 link.
input buffer	input	CHAR(20)	Specifies the name and library of the input buffer that the QOLELINK program will create for this link. The first 10 characters specify the name of the input buffer and the second 10 characters specify the name of an existing library that the input buffer will be created in. Both entries are left-justified. The special values of *LIBL and *CURLIB may be used for the library name. **Note:** A user space object with the same name as the input buffer must not already exist in the specified library.
input buffer descriptor	input	CHAR(20)	Specifies the name and library of the input buffer descriptor that the QOLELINK program will create for this link. The first 10 characters specify the name of the input buffer descriptor and the second 10 characters specify the name of an existing library that the input buffer descriptor will be created in. Both entries are left-justified. The special values of *LIBL and *CURLIB may be used for the library name. **Note:** A user space object with the same name as the input buffer descriptor must not already exist in the specified library.
output buffer	input	CHAR(20)	Specifies the name and library of the output buffer that the QOLELINK program will create for this link. The first 10 characters specify the name of the output buffer and the second 10 characters specify the name of an existing library that the output buffer will be created in. Both entries are left-justified. The special values of *LIBL and *CURLIB may be used for the library name. **Note:** A user space object with the same name as the output buffer must not already exist in the specified library.
output buffer descriptor	input	CHAR(20)	Specifies the name and library of the output buffer descriptor that the QOLELINK program will create for this link. The first 10 characters specify the name of the output buffer descriptor and the second 10 characters specify the name of an existing library that the output buffer descriptor will be created in. Both entries are left-justified. The special values of *LIBL and *CURLIB may be used for the library name. **Note:** A user space object with the same name as the output buffer descriptor must not already exist in the specified library.
key length	input	BINARY(4)	Specifies the key length when using a keyed data queue. Any value between 0 and 256 may be used, where 0 indicates the data queue is not keyed.
key value	input	CHAR(256)	Specifies the key value (left justified) when using a keyed data queue.

Table 2-1 (Page 3 of 3). QOLELINK Parameter List

Parameter	Use	Type	Description
queue name	input	CHAR(20)	Specifies the name and library of the data queue where the enable-complete, disable-complete, permanent-link-failure, and incoming-data entries for this link will be sent. See "Data Queue Entries" on page 6-2 for more information on these data queue entries. The first 10 characters specify the name of an existing data queue and the second 10 characters specify the library in which the data queue is located. Both entries are left-justified. The special values of *LIBL and *CURLIB may be used for the library name.
line description	input	CHAR(10)	Specifies the name of the line description that describes the communications line the link being enabled will use. An existing token-ring, Ethernet, or X.25 line description must be used.
communications handle	input	CHAR(10)	Specifies the name assigned to the link being enabled. Any name complying with system object naming conventions may be used.

Description of Function

The QOLELINK program is called by a user-defined communications application program to enable a link for input/output on a communications line. The communications line, described by the line description parameter, must be a token-ring, Ethernet, or X.25 line. The link being enabled can only be accessed within the job in which the QOLELINK program was called.

Before calling the QOLELINK program to enable a link, the following objects must be configured.

- a token-ring, Ethernet, or X.25 line description
- a data queue

See "Configuring User-Defined Communications Support" on page 6-1 for more information on configuration.

The QOLELINK program will create the input/output buffers and buffer descriptors that will be used for the link being enabled. The user spaces are created with user authority of *PUBLIC and object authority of *EXCLUDE. The user profile of the job calling QOLELINK is assigned *ALL object authority. The network controller description and network device description associated with the link being enabled will also be created if necessary. In addition, the line description, network controller description, and network device description will be varied on if necessary.

When the QOLELINK program returns, the return and reason codes should be examined to determine the status of the link. A successful return and reason code (both zero) indicates the link is being enabled and an enable-complete entry will be sent to the data queue specified on the call to QOLELINK when the enable operation completes. See "Enable-Complete Entry" on page 6-3 for more information on the enable-complete entry. An unsuccessful return and reason code indicates the link could not be enabled and the enable-complete entry will not be sent to the data queue. The following section provides more information on QOLELINK return and reason codes.

Return and Reason Codes

Table 2-2 (Page 1 of 2). Return and Reason Codes for QOLELINK

Return / Reason Code	Meaning	Recovery
0/0	Operation successful, link enabling.	Wait to receive the enable-complete entry from the data queue before doing input/output on this link.
81/9999	Internal system error detected. Escape message CPF91F0 will be sent to the user-defined communications application program when this return and reason code is received.	See messages in the job log for further information. Then, report the problem using the ANZPRB command.
82/1000	User data size not valid for X.25 link.	Correct the X.25 user data size parameter. Then, try the request again.
82/1001	Key length not valid.	Correct the key length parameter. Then, try the request again.
82/1002	Queue name not valid.	Correct the queue name parameter. Then, try the request again.
82/1003	Communications handle not valid.	Correct the communications handle parameter. Then, try the request again.
82/2000	Line description not configured for token-ring, Ethernet, or X.25.	Correct the line description parameter. Then, try the request again.
82/2001	Line description, network controller description, or network device description not in a valid state.	See messages in the job log indicating the affected object and recommended recovery. Do the recovery, and try the request again.
82/2002	Not authorized to the line description or network controller description.	See messages in the job log indicating the affected object and get authorization to it. Then, try the request again.
82/2003	Could not allocate the network device description.	Try the request again. If the problem continues, report the problem using the ANZPRB command.
82/2004	Could not create the network controller description or network device description.	See messages in the job log indicating the affected object and recommended recovery. Do the recovery, and try the request again.
82/2005	Could not vary on the line description, network controller description, or network device description.	See messages in the job log indicating the affected object and recommended recovery. Do the recovery, and try the request again.
82/2006	Line description not found.	Correct the line description parameter. Then, try the request again.
82/2007	Line description damaged.	Delete and recreate the line description. Then, try the request again.
82/2400	An error occurred while creating the input buffer, input buffer descriptor, output buffer, or output buffer descriptor.	See messages in the job log indicating the affected object and recommended recovery. Do the recovery, and try the request again.
82/3000	Communications handle already assigned to another link that is enabled in this job.	Either disable the link that was assigned this communications handle, or correct the communications handle parameter so it does not specify a communications handle that is already assigned to a link enabled in this job. Then, try the request again.

Table 2-2 (Page 2 of 2). Return and Reason Codes for QOLELINK

Return / Reason Code	**Meaning**	**Recovery**
82/3005	Line description already in use by another link that is enabled in this job.	Disable the link that is using this line description. Then, try the request again.

QOLDLINK

```
CALL QOLDLINK(return code,
              reason code,
              communications handle,
              vary option)
```

Parameter List

Table 2-3. QOLDLINK Parameter List

Parameter	Use	Type	Description
return code	output	BINARY(4)	Specifies the recovery action to take. See "Return and Reason Codes" on page 2-9.
reason code	output	BINARY(4)	Specifies the error that occurred. See "Return and Reason Codes" on page 2-9.
communications handle	input	CHAR(10)	Specifies the name of the link to disable. The special value of '*ALL' (left-justified and padded on the right with spaces) may be used to disable all links currently enabled in the job that the user-defined communications application program is running in.
vary option	input	CHAR(1)	Specifies the vary option for the network device description associated with each link being disabled. The valid values are as follows: **X'00'** Do not vary off the network device description. **X'01'** Vary off the network device description.

Description of Function

The QOLDLINK program is called by a user-defined communications application program to disable one or all links that are currently enabled in the job in which the user-defined communications application program is running. When a link is disabled, all system resources used by that link are released, the input/output buffers and descriptors for that link are deleted, and input/output on that link is no longer possible.

In addition to a user-defined communications application program explicitly disabling a link by calling the QOLDLINK program, user-defined communications support will implicitly disable a link in the following cases:

- when the network device associated with an enabled link is varied off from the job in which it was enabled
- when a job ends in which one or more links were enabled
- when the user-defined communications application program that enabled the link ends abnormally
- when the Reclaim Resource (RCLRSC) command is used

For each link that is successfully disabled, either explicitly or implicitly, the disable-complete entry will be sent to the data queue that was specified on the call to the QOLELINK program when the link was enabled. See "Disable-Complete Entry" on page 6-3 for the format of the disable-complete entry.

Return and Reason Codes

Table 2-4. Return and Reason Codes for QOLDLINK

Return / Reason Code	**Meaning**	**Recovery**
0/0	Operation successful.	Continue processing.
83/1004	Vary option not valid.	Correct the vary option parameter. Then, try the request again.
83/3001	Link not enabled.	Correct the communications handle parameter. Then, try the request again.

QOLSETF

```
CALL QOLSETF(return code,
             reason code,
             error offset,
             communications handle)
```

Parameter List

Table 2-5. QOLSETF Parameter List

Parameter	Use	Type	Description
return code	output	BINARY(4)	Specifies the recovery action to take. See "Return and Reason Codes" on page 2-15.
reason code	output	BINARY(4)	Specifies the error that occurred. See "Return and Reason Codes" on page 2-15.
error offset	output	BINARY(4)	Specifies the offset from the top of the output buffer to the incorrect filter header data or to the incorrect filter in the filter list. The content of this parameter is only valid for 83/1999 and 83/3003 return/reason codes.
communications handle	input	CHAR(10)	Specifies the name of the link on which to perform the filter operation.

Description of Function

The QOLSETF program is called by a user-defined communications application program to activate and/or deactivate one or more filters for a link that is currently enabled in the job in which the user-defined communications application program is running. The required filter information must be provided by the user-defined communications application program in the output buffer that was created when the link was enabled. The output buffer descriptor is not used. See "Format of Filter Information" on page 2-11 for details on the format of the filter information in the output buffer.

Filters contain inbound routing information that user-defined communications support uses to route incoming data to a link that is enabled by a user-defined communications application program. The incoming data that is routed depends on the type of communications line the link is using. On an X.25 communications line, the incoming data is an incoming switched virtual circuit (SVC) call. On a token-ring or Ethernet communications line, the incoming data is the actual data frame.

How incoming data is routed to a link is determined by the type of filters activated for that link[1]. For links using a token-ring or Ethernet communications line, there are three types of filters. They are listed below from most to least restrictive:

- destination service access point (DSAP), source service access point (SSAP), optional frame type, and sending adapter address

[1] All active filters for a link must be of the same type.

- DSAP, SSAP, and optional frame type
- DSAP

For links using an X.25 communications line, there are two types of filters. They are listed below from most to least restrictive:

- Protocol identifier (PID) and calling data terminal equipment (DTE) address

  The AS/400 system treats the first byte of call-user data in an X.25 call request packet as the PID.

- PID

The order of checking filters when multiple links are using the same communications line is from most to least restrictive. For example, suppose two user-defined communications application programs (application program A and B) in different jobs each have a link enabled that use the same token-ring communications line. Further suppose that application program A has activated a filter on DSAP X′22′ and application program B has activated a filter on DSAP X′22′ and SSAP X′22′. If a data frame comes in with a DSAP of X′22′ and an SSAP of X′22′, application program B will receive the frame. If a data frame comes in with a DSAP of X′22′ and an SSAP not equal to X′22′, application program A will receive the frame.

The Display Connection Status (DSPCNNSTS) command can be used to display the inbound routing information of all filters currently active for a link. See the *CL Reference* for more information on the DSPCNNSTS command.

Format of Filter Information

All filter information must be provided by the user-defined communications application program in the output buffer that was created when the link was enabled. The output buffer should be treated as one large space with the size equal to the number of data units created for the output buffer multiplied by the size of each data unit. This information was returned by the QOLELINK program when the link was enabled.

The filter information in the output buffer is made up of two parts. The first portion starts at offset 0 from the top of the output buffer and contains filter header data. The second portion of the filter information starts immediately after the filter header data in the output buffer and contains the filters that make up the filter list.

Table 2-6 (Page 1 of 2). Filter Header Data

Field	Type	Description
function	CHAR(1)	Specifies the filter function to perform. The valid values are as follows: **X′00′** Deactivate all filters that are currently active for this link and activate the filters specified in the filter list for this link. **X′01′** Activate the filters specified in the filter list for this link. All filters currently active for this link will remain active. **X′02′** Deactivate the filters specified in the filter list that are currently active for this link.

Table 2-6 (Page 2 of 2). Filter Header Data

Field	Type	Description
filter type	CHAR(1)	Specifies the type of the filters in the filter list. All filters in the filter list must be of this type. In addition, this must be the same type as the filters currently active for this link, if any. The valid values are as follows: **X'00'** PID. This filter type is only applicable for links using an X.25 communications line and only applies to incoming SVC calls. **X'01'** PID and calling DTE address. This filter type is only applicable for links using an X.25 communications line and only applies to incoming SVC calls. **X'02'** DSAP. This filter type is only applicable for links using a token-ring or Ethernet communications line. **X'03'** DSAP, SSAP, and optional frame type. This filter type is only applicable for links using a token-ring or Ethernet communications line. **X'04'** DSAP, SSAP, optional frame type, and sending adapter address. This filter type is only applicable for links using a token-ring or Ethernet communications line. **Note:** The filter type field must be set even if there are no filters in the filter list.
number of filters	BINARY(2)	Specifies the number of filters in the filter list. Any value between 0 and 256 may be used. **Note:** The maximum number of filters that can be specified in the filter list is also limited by the total size of the output buffer which may accommodate less than 256 filters.
filter length	BINARY(2)	Specifies the length of each filter in the filter list. This value must be 16 for filter types X'00' and X'01', and 14 for filter types X'02', X'03', and X'04'. **Note:** The filter length field must be set even if there are no filters in the filter list.

The format of each filter in the filter list is described below. All filters in the filter list must be contiguous with each other and be of the type specified in the filter type field in the filter header data.

X.25 Filters (Filter Types X'00' and X'01')

Table 2-7 (Page 1 of 2). Filter Types X'00' and X'01'

Field	Type	Description
PID length	CHAR(1)	Specifies the length of the PID on which to route incoming calls. The valid values are as follows: **X'00'** Route incoming calls with no PID specified. That is, with no call user data in the call request packet. **X'01'** Route incoming calls with the PID being treated as the first byte of call user data in the call request packet.

Table 2-7 (Page 2 of 2). Filter Types X'00' and X'01'

Field	Type	Description
PID	CHAR(1)	Specifies the PID on which to route incoming calls. This should be set to X'00' when the PID length field is set to X'00'. Otherwise, any value may be used. **Note:** Care should be taken when setting the PID field to an SNA PID (X'C3', X'C6', X'CB', X'CE'), Asynchronous PID (X'01', X'C0'), or TCP/IP PID (X'CC'). See "AS/400 System X.25 Call Control" on page 3-29 for more information.
calling DTE address length	CHAR(1)	Specifies, in hexadecimal, the number of binary coded decimal (BCD) digits in the calling DTE address on which to route incoming calls. The valid values are as follows: **X'00'** For filter type X'00'. **X'01' - X'0F'** For filter type X'01' when extended network addressing is not configured in the line description. See "QOLQLIND" on page 2-57 to determine if extended network addressing is configured for this line. **X'01' - X'11'** For filter type X'01' when extended network addressing is configured in the line description. See "QOLQLIND" on page 2-57 to determine if extended network addressing is configured for this line.
calling DTE address	CHAR(12)	Specifies, in binary coded decimal (BCD), the calling DTE address on which to route incoming calls. This should be set to BCD zeros when the calling DTE address length field is set to X'00'. Otherwise, any valid DTE address left-justified and padded on the right with BCD zeros may be used.
additional routing data	CHAR(1)	Specifies additional data on which to route incoming calls. This field is applicable for all X.25 filter types and is bit-sensitive with bit 0 (leftmost bit) defined for reverse charging options and bit 1 defined for fast select options. The remaining bits are undefined and should be set off ('0'B). The valid values for bit 0 are as follows: **'0'B** Accept reverse charging. **'1'B** Do not accept reverse charging. The valid values for bit 1 are as follows: **'0'B** Accept fast select. **'1'B** Do not accept fast select. For example, consider the following values for the additional routing data field: **X'00'** Accept reverse charging and accept fast select. **X'40'** Accept reverse charging and do not accept fast select. **X'80'** Do not accept reverse charging and accept fast select. **X'C0'** Do not accept reverse charging and do not accept fast select.

Token-ring and Ethernet Filters (Filter Types X'02', X'03' and X'04')

Table 2-8. Filter Types X'02', X'03' and X'04'

Field	**Type**	**Description**
DSAP address length	CHAR(1)	Specifies the length of the DSAP address on which to route incoming frames. This must be set to X'01'.
DSAP address	CHAR(1)	Specifies the DSAP address on which to route incoming frames. The DSAP address is the service access point on which the incoming frame arrived. Any service access point configured in the token-ring or Ethernet line description as *NONSNA may be used. **Note:** The Ethernet Version 2 standard does not define a DSAP address in an Ethernet Version 2 frame. Therefore, to receive Ethernet Version 2 frames, a null DSAP address (X'00') must be specified in the DSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.
SSAP address length	CHAR(1)	Specifies the length of the SSAP address on which to route incoming frames. The valid values are as follows: **X'00'** For filter type X'02'. **X'01'** For filter types X'03' and X'04'.
SSAP address	CHAR(1)	Specifies the SSAP address on which to route incoming frames. The SSAP address is the service access point on which the incoming frame was sent. The valid values are as follows: **X'00'** For filter type X'02'. **X'00' - X'FF'.** For filter types X'03' and X'04'. **Note:** The Ethernet Version 2 standard does not define an SSAP address in an Ethernet Version 2 frame. Therefore, to receive Ethernet Version 2 frames, a null SSAP address (X'00') must be specified in the SSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.
frame type length	CHAR(1)	Specifies the length of the frame type on which to route incoming frames. The valid values are as follows: **X'00'** For filter type X'02'. Also for filter types X'03' and X'04' when the DSAP address and SSAP address fields are not both set to X'00'. **X'00' or X'02'** For filter types X'03' and X'04' when the 'DSAP address' and SSAP address fields are both set to X'00'.
frame type	CHAR(2)	Specifies the frame type on which to route incoming frames. The frame type is defined in an Ethernet Version 2 frame to indicate the upper layer protocol being used. This must be set to X'0000' when the frame type length field is set to X'00'. Otherwise, any value except X'80D5' (SNA) may be used, but should be in the range of X'05DD' - X'FFFF'.
sending adapter address length	CHAR(1)	Specifies, in hexadecimal, the length of the sending adapter address on which to route incoming frames. The valid values are as follows: **X'00'** For filter types X'02' and X'03'. **X'06'** For filter type X'04'.
sending adapter address	CHAR(6)	Specifies, in packed form, the sending adapter address on which to route incoming frames. This must be set to X'000000000000' when the sending adapter address length field is set to X'00'. Otherwise, any valid adapter address may be used.

General Filter Rules

The following is a list of rules for activating and deactivating filters:

- all active filters for a link must be of the same type
- a link can have a maximum of 256 active filters
- the maximum number of filters that can be specified in the filter list can be no more than 256, and may be less, depending on the size of the output buffer
- a request to activate a filter for a link that already has the same filter active will be successful, but the filter will only be activated once
- a request to deactivate a filter for a link that has no such filter active will be successful
- if the return and reason code from the QOLSETF program is not 0/0, none of the specified filters were activated or deactivated
- once a filter is activated, it will remain active until one of the following occurs:
  - it is deactivated by explicitly calling the QOLSETF program
  - the link that the filter was active for is disabled

Return and Reason Codes

Table 2-9 (Page 1 of 2). Return and Reason Codes for QOLSETF

Return / Reason Code	**Meaning**	**Recovery**
0/0	Operation successful.	Continue processing.
80/2200	Data queue error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Then correct the error, enable the link, and try the request again.
80/2401	Output buffer error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Then correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the user-defined communications application program by escape message CPF91F0 or CPF91F1. However, the user-defined communications application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the user-defined communications application program, then report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Then correct the condition, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the user-defined communications application program when this return and reason code is received.	See messages in the job log for further information. Then, report the problem using the ANZPRB command.

Table 2-9 (Page 2 of 2). Return and Reason Codes for QOLSETF

Return / Reason Code	Meaning	Recovery
83/1998	The size of the output buffer is not large enough for the specified number of filters.	Reduce the number of filters in the filter list so that the size of the filter list plus the size of the filter header data is less than or equal to the size of the output buffer. Try the request again.
83/1999	Incorrect filter header data or incorrect filter in the filter list. If the filter header data is incorrect, the error offset parameter will point to the field in error. If a filter in the filter list is incorrect, the error offset parameter will point to the beginning of the incorrect filter.	Correct the incorrect filter header data or the incorrect filter in the filter list. Try the request again.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3003	One of the following is true of a filter in the filter list. The error offset parameter will point to the beginning of the offending filter. • the filter is already activated by another job using the same communications line • the service access point, specified in the DSAP address field of the filter, is not configured in the token-ring or Ethernet line description • the DSAP address field of the filter contains the null DSAP address (X'00'), but the Ethernet Standard (ETHSTD) parameter in the Ethernet line description is not configured as *ETHV2 or *ALL • the service access point, specified in the DSAP address field of the filter, is configured in the token-ring or Ethernet line description for SNA use only (*SNA)	Do one of the following, and try the request again: • end the job that has already activated the filter • configure the service access point in the token-ring or Ethernet line description • delete the Ethernet line description, and create another Ethernet line description specifying *ETHV2 or *ALL in the Ethernet Standard (ETHSTD) parameter • change the service access point in the token-ring or Ethernet line description to non-SNA use (*NONSNA)
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue. If the link was successfully enabled, try the request again.
83/3200	All resources are currently in use by asynchronous operations that have not yet completed. **Note:** This return and reason code is only possible for links using an X.25 communications line. See "QOLSEND" on page 2-17 for more information.	Wait for at least one of the asynchronous operations to complete. Notification of completion of these operations will be received from the QOLRECV program. Try the request again.
83/4001	Link failure, system starting error recovery for this link.	Wait for the link to recover. Try the request again.

QOLSEND

```
CALL QOLSEND(return code,
             reason code,
             diagnostic data,
             new provider connection end point ID,
             new user connection end point ID,
             existing provider connection end point ID,
             communications handle,
             operation,
             number of data units)
```

Parameter List

Table 2-10 (Page 1 of 2). QOLSEND Parameter List

Parameter	Use	Type	Description
return code	output	BINARY(4)	Specifies the recovery action to take. See "Return and Reason Codes" on page 2-35.
reason code	output	BINARY(4)	Specifies the error that occurred. See "Return and Reason Codes" on page 2-35.
diagnostic data	output	CHAR(40)	Specifies additional diagnostic data. See "Format of Diagnostic Data Parameter" on page 2-18 for more information. The content of this parameter is only valid when the operation parameter is set to X'0000' or X'B400'.
new provider connection end point ID	output	BINARY(4)	Specifies the provider connection end point (PCEP) ID for the connection that is to be established. This identifier must be used on all subsequent calls to the QOLSEND program for this connection. The content of this parameter is only valid for links using an X.25 communications line and when the operation parameter is set to X'B000'.
new user connection end point ID	input	BINARY(4)	Specifies the user connection end point (UCEP) ID for the connection that is to be established. This is the identifier on which all incoming data for this connection will be received. Any numeric value except zero should be used. See "QOLRECV" on page 2-40 for more information. The content of this parameter is only valid for links using an X.25 communications line and when the operation parameter is set to X'B000' or X'B400'.
existing provider connection end point ID	input	BINARY(4)	Specifies the PCEP ID for the connection on which this operation will be performed. For links using a token-ring or Ethernet communications line, the content of this parameter must always be set to 1. For links using an X.25 communications line, the content of this parameter is only valid when the operation parameter is set to X'0000', X'B100', X'B400', or X'BF00'. It must contain the PCEP ID that was returned in the new provider connection end point ID parameter from the call to the QOLSEND program with operation X'B000', or the PCEP ID that was returned in the new provider connection end point ID parameter from the call to the QOLRECV program with operation X'B201' (incoming call). See "QOLRECV" on page 2-40 for more information on receiving X.25 calls.

Table 2-10 (Page 2 of 2). QOLSEND Parameter List

Parameter	Use	Type	Description
communications handle	input	CHAR(10)	Specifies the name of the link on which to perform the output operation.
operation	input	CHAR(2)	Specifies the type of output operation to perform. The valid values are as follows: **X'0000'** Send data. **X'B000'** Send call request packet (SVC) or open PVC connection. This operation is only valid for links using an X.25 communications line. **X'B100'** Send clear packet (SVC) or close PVC connection. This operation is only valid for links using an X.25 communications line. **X'B400'** Send call accept packet (SVC). This operation is only valid for links using an X.25 communications line. **X'BF00'** Send reset request packet or reset confirmation packet (SVC or PVC). This operation is only valid for links using an X.25 communications line.
number of data units	input	BINARY(4)	Specifies the number of data units in the output buffer that contain data. Any value between 1 and the number of data units created in the output buffer may be used. The content of this parameter is only valid when the operation parameter is set to X'0000'. **Note:** The number of data units created in the output buffer was returned in the data units created parameter on the call to the QOLELINK program. See "QOLELINK" on page 2-3 for more information.

Format of Diagnostic Data Parameter

The format of the diagnostic data parameter is shown below. The contents of the fields within this parameter are only valid on X'0000' and X'B400' operations for the indicated return and reason codes.

Table 2-11 (Page 1 of 3). Diagnostic Data Parameter

Field	Type	Description
reserved	CHAR(2)	Not used.
error code	CHAR(4)	Specifies hexadecimal diagnostic information that can be used to determine recovery actions. See "Error Codes" on page 5-11 for more information. The content of this field is only valid for 83/4001, 83/4002, and 83/4003 return/reason codes.
time stamp	CHAR(8)	Specifies the time the error occurred. The content of this field is only valid for 83/4001, 83/4002, and 83/4003 return/reason codes.

Table 2-11 (Page 2 of 3). Diagnostic Data Parameter

Field	Type	Description
error log identifier	CHAR(4)	Specifies the hexadecimal identifier that can be used for locating error information in the error log. The content of this field is only valid for 83/4001, 83/4002, and 83/4003 return/reason codes.
reserved	CHAR(10)	Not used.
indicators	CHAR(1)	Specifies indicators the user-defined communications application program can use for diagnosing a potential error condition. This is a bit sensitive field. The valid values for bit 0 (leftmost bit) are as follows: **'0'B** Either there is no message in the QSYSOPR message queue, or there is a message and it does not have the capability to run problem analysis report (PAR) to determine the cause of the error. **'1'B** There is a message in the QSYSOPR message queue for this error, and it does have the capability to run problem analysis report (PAR) to determine the cause of the error. The valid values for bit 1 are as follows: **'0'B** The line error can be retried. **'1'B** The line error cannot be retried. The valid values for bit 2 are as follows: **'0'B** The cause and diagnostic codes fields are not valid. **'1'B** The cause and diagnostic codes fields are valid. The valid values for bit 3 are as follows: **'0'B** The error has not been reported to the system operator message queue. **'1'B** The error has been reported to the system operator message queue. For example, consider the following values for the indicators field: **X'20'** A condition has caused X.25 cause and diagnostic codes to be passed to the application. This information can determine the cause of the condition. **X'50'** An error has occurred and been reported to the QSYSOPR message queue. The error cannot be retried. **X'F0'** An error has occurred and been reported to the QSYSOPR message queue. The error cannot be retried, and has X.25 cause and diagnostic codes associated with it. Also a problem analysis report can be generated to determine the probable cause. The content of this field is only valid for 83/4001, 83/4002, 83/3202 and 83/4003 return/reason codes.
X.25 cause code	CHAR(1)	Specifies additional information on the condition reported. See the *X.25 Network Guide* for interpreting the values of this field. The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes.
X.25 diagnostic code	CHAR(1)	Specifies additional information on the condition reported. See the *X.25 Network Guide* for interpreting the values of this field. The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes.

Table 2-11 (Page 3 of 3). Diagnostic Data Parameter

Field	Type	Description
reserved	CHAR(1)	Not used.
error offset	BINARY(4)	Specifies the offset from the top of the output buffer to the incorrect data in the output buffer. The content of this field is only valid for a 83/1999 return/reason code.
reserved	CHAR(4)	Not used.

Description of Function

The QOLSEND program is called by a user-defined communications application program to perform output on a link that is currently enabled in the job in which the user-defined communications application program is running. The type of output operation to perform is specified in the operation parameter. The data associated with the output operation must be provided by the user-defined communications application program in the output buffer that was created when the link was enabled. For X'0000' operations, the user-defined communications application program must also provide a description of that data in the output buffer descriptor that was created when the link was enabled.

The types of output operations that can be performed on a link depend on the type of communications line that the link is using. See "LAN Output Operations" for more information on output operations that are supported on links using a token-ring or Ethernet communications line. See "X.25 SVC and PVC Output Operations" on page 2-22 for more information on output operations that are supported on links using an X.25 communications line.

LAN Output Operations

The only supported output operation on links using a token-ring or Ethernet communications line is X'0000' (send user data). For each data frame to be sent on the network, the user-defined communications application program must provide the following information:

- logical link control (LLC) information, optional routing information, and user data in the next data unit of the output buffer, starting with the first data unit
- a description, in the corresponding element of the output buffer descriptor, of the information in that data unit.

For example, suppose a user-defined communications application program wants to send two data frames. The information for the first frame would be placed in first data unit of the output buffer and described in the first element of the output buffer descriptor. The information for the second frame would be placed in the second data unit of the output buffer and described in the second element of the output buffer descriptor. The number of data units parameter on the call to the QOLSEND program would be set to 2.

Note: The X'0000' operation is synchronous. Control will not return from the QOLSEND program until the operation completes.

Data Unit Format - LAN Operation X'0000': Each data frame to be sent on the network corresponds to a data unit in the output buffer. The information in each of these data units is made up of LLC information, optional routing data, and user data.

The LLC information starts at offset 0 from the top of the data unit. The routing information (if any) starts immediately after the LLC information and the user data starts immediately after the routing information. If there isn't any routing information, the user data starts immediately after the LLC information.

Table 2-12 shows the format of the LLC information.

Table 2-12 (Page 1 of 2). Format of the LLC Information

Field	**Type**	**Description**
length of LLC information	BINARY(2)	Specifies the length of the LLC information in the data unit. This must be set to 16.
destination adapter address	CHAR(6)	Specifies, in packed form, the adapter address to which this data frame will be sent. **Note:** Because user-defined communications support only allows connectionless service over LANs, it is not necessary for all frames being sent on a single output operation to have the same destination adapter address.
DSAP address	CHAR(1)	Specifies the service access point on which the destination system will receive this frame. Any value may be used. **Note:** The Ethernet Version 2 standard does not define a DSAP address in an Ethernet Version 2 frame. Therefore, to send Ethernet Version 2 frames, a null DSAP address (X'00') must be specified in the DSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.
SSAP address	CHAR(1)	Specifies the service access point on which the AS/400 system will send this frame. Any service access point configured in the token ring or Ethernet line description may be used. **Note:** The Ethernet Version 2 standard does not define an SSAP address in an Ethernet version 2 frame. Therefore, to send Ethernet Version 2 frames, a null SSAP address (X'00') must be specified in the SSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.
access control	CHAR(1)	Specifies outbound frame priority and is mapped to the access priority bits in the access control field of 802.5 frames. For links using a token-ring communications line, any value between X'00' and X'07' may be used, where X'00' is the lowest priority and X'07' is the highest priority. For links using an Ethernet communications line, the content of this field is not applicable and must be set to X'00'.
priority control	CHAR(1)	Specifies how to interpret the value set in the access control field. For links using a token-ring communications line, the valid values are as follows: **X'00'** Use any priority less than or equal to the value set in the access control field. **X'01'** Use the priority exactly equal to the value set in the access control field. **X'FF'** Use the AS/400 system default priority. For links using an Ethernet communications line, the content of this field is not applicable and must be set to X'00'.

Table 2-12 (Page 2 of 2). Format of the LLC Information

Field	Type	Description
length of routing information	BINARY(2)	Specifies the length of the routing information in the data unit. For links using a token-ring communications line, any value between 0 and 18 may be used, where 0 indicates that there is no routing information. For links using an Ethernet communications line, the content of this field is not applicable and must be set to 0 indicating that there is no routing information.
length of user data	BINARY(2)	Specifies the length of the user data in the data unit. This must be less than or equal to the maximum frame size allowed on the service access point specified in the SSAP address field. See "QOLQLIND" on page 2-57 to determine the maximum frame size allowed on the service access point specified in the SSAP address field. For Ethernet Version 2 frames, this must be at least 48 and not more than 1502 (including 2 bytes for the Ethernet type field). **Note:** Ethernet 802.3 frames will be padded when the user data is less than 48 bytes.

Output Buffer Descriptor Element Format - LAN Operation X'0000': The information specified in each data unit of the output buffer must be described in the corresponding element of the output buffer descriptor. Table 2-13 shows the format of each element in the output buffer descriptor.

Table 2-13. Format of an Element in the Output Buffer Descriptor

Field	Type	Description
length	BINARY(2)	Specifies the number of bytes of information in the corresponding data unit of the output buffer. This must be equal to the length of the LLC information plus the length of the routing information plus the length of the user data specified in Table 2-12 on page 2-21.
reserved	CHAR(30)	Not used.

X.25 SVC and PVC Output Operations

Table 2-14 shows the output operations that are supported on links using an X.25 communications line.

Table 2-14 (Page 1 of 2). X.25 SVC and PVC Output Operations

Operation	Meaning
X'0000'	Send user data (SVC or PVC). **Note:** This is a synchronous operation. Control will not return from the QOLSEND program until the operation completes.
X'B000'	Send a call request packet (SVC) or open the PVC connection. **Note:** This is an asynchronous operation. Notification of the completion of this operation will be returned from the QOLRECV program with operation X'B001' only after control returns from the QOLSEND program with a 0/0 return and reason code. See "QOLRECV" on page 2-40 for more information.

Table 2-14 (Page 2 of 2). X.25 SVC and PVC Output Operations

Operation	Meaning
X'B100'	Send a clear packet (SVC) or close the PVC connection. **Note:** This is an asynchronous operation. Notification of the completion of this operation will be returned from the QOLRECV program with operation X'B101' only after control returns from the QOLSEND program with a 0/0 return and reason code. See "QOLRECV" on page 2-40 for more information.
X'B400'	Send a call accept packet (SVC only). **Note:** This is a synchronous operation. Control will not return from the QOLSEND program until the operation completes.
X'BF00'	Send a reset request or reset confirmation packet (SVC or PVC). **Note:** This is an asynchronous operation. Notification of the completion of this operation will be returned from the QOLRECV program with operation X'BF01' only after control returns from the QOLSEND program with a 0/0 return and reason code. See "QOLRECV" on page 2-40 for more information.
Note: The maximum number of outstanding asynchronous operations (notification of completion not yet received from the QOLRECV program) is five. All calls made to the QOLSEND program or QOLSETF program under this condition will be rejected with a return and reason code of 83/3200.	

X.25 Operation X'0000'

This operation is used to send user data on an SVC or PVC X.25 connection. The user-defined communications application program must provide the following information:

- user data in the next data unit of the output buffer, starting with the first data unit
- a description, in the corresponding element of the output buffer descriptor, of the user data in that data unit.

For example, suppose a user-defined communications application program wants to send two data units of user data. The first portion of the user data would be placed in first data unit of the output buffer and described in the first element of the output buffer descriptor. The second portion of the user data would be placed in the second data unit of the output buffer and described in the second element of the output buffer descriptor. The number of data units parameter on the call to the QOLSEND program would be set to 2.

User-defined communications support will automatically fragment the user data in each data unit into one or more appropriately sized X.25 packets based on the negotiated transmit packet size for the connection. All packets constructed for a data unit, except for the last (or only) packet, will always have the X.25 more data bit (M-bit) set on. See "Output Buffer Descriptor Element Format - X.25 Operation X'0000'" for more information on how to set the X.25 M-bit on or off in the last (or only) packet constructed for a data unit.

Data Unit Format - X.25 Operation X'0000': Each data unit in the output buffer consists solely of user data and starts offset 0 from the top of the data unit.

Output Buffer Descriptor Element Format - X.25 Operation X'0000': The user data specified in each data unit of the output buffer must be described in the corresponding element of the output buffer descriptor.

Table 2-15 shows the format of each element in the output buffer descriptor.

Table 2-15. Format of an Element in the Output Buffer Descriptor

Field	Type	Description
length	BINARY(2)	Specifies the number of bytes of user data in the corresponding data unit of the output buffer. This must always be less than or equal to the X.25 user data size parameter that was specified on the call to the QOLELINK program when the link was enabled. See "QOLELINK" on page 2-3 for more information.
more data indicator	CHAR(1)	Specifies if the X.25 more data bit (M-bit) should be set on or off in the last (or only) X.25 packet constructed for the corresponding data unit. The valid values are as follows: **X'00'** Set the M-bit off in the last (or only) X.25 packet constructed for the corresponding data unit. **X'01'** Set the M-bit on in the last (or only) X.25 packet constructed for the corresponding data unit. **Note:** When this value is selected, the length field must be set to a multiple of the negotiated transmit packet size for the connection.
qualified data indicator	CHAR(1)	Specifies if the X.25 qualifier bit (Q-bit) should be set on or off in all X.25 packets constructed for the corresponding data unit. The valid values are as follows: **X'00'** Set the Q-bit off in all X.25 packets constructed for the corresponding data unit. **X'01'** Set the Q-bit on in all X.25 packets constructed for the corresponding data unit.
interrupt packet indicator	CHAR(1)	Specifies if the user data in the corresponding data unit should be sent in an X.25 interrupt packet. The valid values are as follows: **X'00'** Send the user data in the corresponding data unit in one or more X.25 data packets. **X'01'** Send the user data in the corresponding data unit in an X.25 interrupt packet. An interrupt packet causes the data to be expedited. **Note:** When this value is selected, the length field must be set to a value between 1 and 32, and number of data units parameter on the call to the QOLSEND program must be set to 1. Also, the contents of the more data indicator, qualified data indicator, and delivery confirmation indicator fields are ignored.
delivery confirmation indicator	CHAR(1)	Specifies if the X.25 delivery confirmation bit (D-bit) should be set on or off in all X.25 packets constructed for the corresponding data unit. The valid values are as follows: **X'00'** Set the D-bit off in all X.25 packets constructed for the corresponding data unit. **X'01'** Set the D-bit on in all X.25 packets constructed for the corresponding data unit. **Note:** The AS/400 system does not fully support delivery confirmation when sending user data. Confirmation is from the local data circuit equipment (DCE).
reserved	CHAR(26)	Not used.

X.25 Operation X'B000'

This operation is used to either initiate an SVC call or to open a PVC connection. The user-defined communications application program must provide the data for this operation in the first data unit of the output buffer. The output buffer descriptor is not used.

The format of the data required for the X'B000' operation depends on whether it is used to initiate an SVC call or to open a PVC connection. Each format will be explained below.

Note: When initiating an SVC call, the AS/400 system will choose an available SVC to use. The logical channel identifier of the SVC that was chosen will be returned when notification of the completion of X'B000' is received from the QOLRECV program (operation X'B001'). See "QOLRECV" on page 2-40 for more information.

Data Unit Format - X.25 Operation X'B000' (Initiate an SVC Call): The data for this operation starts at offset 0 from the top of the first data unit in the output buffer.

Table 2-16 shows the format of the data required for the X'B000' operation when initiating an SVC call.

Table 2-16 (Page 1 of 4). Format of Data for X'B000' Operation (Initiate an SVC Call)

Field	Type	Description
reserved	CHAR(1)	This field must be set to X'02'.
reserved	CHAR(3)	This field must be set to hexadecimal zeros (X'000000').
transmit packet size	BINARY(2)	Specifies the requested transmit packet size for this connection. The valid values are 64, 128, 256, 512, and 1024. The value specified must be less than or equal to the transmit maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the transmit default packet size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the transmit maximum packet size and the transmit default packet size configured for this line.
transmit window size	BINARY(2)	Specifies the requested transmit window size for this connection. The valid values are as follows: **1 - 7** When modulus 8 is configured for this line. **1 - 15** When modulus 128 is configured for this line. **X'FFFF'** Use the transmit default window size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the modulus value and the transmit default window size configured for this line.
receive packet size	BINARY(2)	Specifies the requested receive packet size for this connection. The valid values are 64, 128, 256, 512, and 1024. The value specified must be less than or equal to the receive maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the receive default packet size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the receive maximum packet size and the receive default packet size configured for this line.

Table 2-16 (Page 2 of 4). *Format of Data for X'B000' Operation (Initiate an SVC Call)*

Field	Type	Description
receive window size	BINARY(2)	Specifies the requested receive window size for this connection. The valid values are as follows: **1 - 7** When modulus 8 is configured for this line. **1 - 15** When modulus 128 is configured for this line. **X'FFFF'** Use the receive default window size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the modulus value and the receive default window size configured for this line.
reserved	CHAR(7)	Not used. This field should be set to hexadecimal zeros.
DTE address length	BINARY(1)	Specifies the number of binary coded decimal (BCD) digits in the DTE address to call. The valid values are as follows: **1 - 15** When extended network addressing is not configured for this line. **1 - 17** When extended network addressing is configured in the line description. See "QOLQLIND" on page 2-57 to determine if extended network addressing is configured for this line.
DTE address	CHAR(16)	Specifies, in binary coded decimal (BCD), the DTE address to call. The address must be left justified and padded on the right with BCD zeros.
reserved	CHAR(8)	Not used. This field should be set to hexadecimal zeros.
delivery confir-mation support	CHAR(1)	Specifies if the X.25 delivery confirmation bit (D-bit) should be set on or off in the call request packet. The valid values are as follows: **X'00'** Set the D-bit off in the call request packet. **X'01'** Set the D-bit on in the call request packet.
reserved	CHAR(7)	Not used. This field should be set to hexadecimal zeros.
closed user group indicator	CHAR(1)	Specifies if the closed user group (CUG) identifier should be included in the call packet. The valid values are as follows: **X'00'** Do not include the CUG identifier in the call packet. **X'01'** Include the CUG identifier in the call packet.
closed user group identifier	CHAR(1)	Specifies the CUG identifier to be included in the call packet. The valid values are as follows: **X'00'** When the closed user group indicator field is set to X'00' **X'00' - X'99'** When the closed user group indicator field is set to X'01'
reverse charging indicator	CHAR(1)	Specifies reverse charging options. The valid values are as follows: **X'00'** Do not request reverse charging. **X'01'** Request reverse charging.
fast select indi-cator	CHAR(1)	Specifies fast select options. The valid values are as follows: **X'00'** Do not request fast select. **X'01'** Request fast select with restriction. **X'02'** Request fast select without restriction.

Table 2-16 (Page 3 of 4). Format of Data for X'B000' Operation (Initiate an SVC Call)

Field	Type	Description
X.25 facilities length	BINARY(1)	Specifies the number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be used. **Note:** The AS/400 system codes the closed user group, reverse charging, and fast select facilities in the X.25 facilities field, if the user requested them in the above fields. Additionally, if the network user identification parameter (NETUSRID) is specified in the line description, the network user identification (NUI) facility is coded in the field, following the other additional facilities, if present. Finally, if the packet and window size values specified are different than the network default, the facilities containing these values are coded in the field as well. The system will update the X.25 facilities length field appropriately for each facility to which the AS/400 system adds the X.25 facilities field. This length cannot exceed 109 bytes.
X.25 facilities	CHAR(109)	Specifies additional X.25 facilities data requested. **Note:** The application programmer should not code the facilities for NUI, fast select, reverse charging, closed user group, packet size, or window size in this field. By doing so, this field could contain duplicate facilities, which may not be consistently supported by all X.25 networks.
reserved	CHAR(48)	Not used. This field should be set to hexadecimal zeros.
call user data length	BINARY(2)	Specifies the number of bytes of data in the call user data field. The valid values are as follows: **0 - 16** When the fast select indicator field is set to X'00'. **0 - 128** When the fast select indicator field is set to X'01' or X'02'.
call user data	CHAR(128)	Specifies the call user data.
reserved	CHAR(128)	Not used. This field should be set to hexadecimal zeros.
control information	CHAR(1)	Specifies control information for this connection. This is a bit-sensitive field with bit 0 (leftmost bit) defined for reset support. The remaining bits are undefined and should be set off ('0'B). The valid values for bit 0 are as follows: **'0'B** Resets are not supported on this connection. When this value is selected, the X'BF00' output operation will not be valid on this connection. Also, a reset indication packet received on this connection will cause the connection to be ended. **'1'B** Resets are supported on this connection. When this value is selected, the X'BF00' output operation will be valid on this connection. Also, the user-defined communications application program will be required to handle reset indications received on this connection. For example, consider the following values for the control information field: **X'00'** Resets are not supported on this connection. **X'80'** Resets are supported on this connection.
reserved	CHAR(3)	Not used. This field should be set to hexadecimal zeros.

Table 2-16 (Page 4 of 4). Format of Data for X'B000' Operation (Initiate an SVC Call)

Field	Type	Description
maximum data unit assembly size	BINARY(4)	Specifies the maximum number of bytes of user data that can be received in a complete X.25 packet sequence on this connection. If this limit is exceeded, the connection will be ended. Any value between 1024 and 32767 may be used, and should be set to the greatest value that the application will support. **Notes:** 1. If the number of bytes of user data received in a complete X.25 packet sequence is more than can fit into one data unit of the input buffer, the more data indicator field in the corresponding element of the input buffer descriptor will be set to X'01' and the remaining user data will be filled in the next data unit. See "QOLRECV" on page 2-40 for more information. 2. There is no limitation on the number of bytes of user data that can be sent in a complete X.25 packet sequence. However, the QOLSEND program may need to called more than once.
automatic flow control	BINARY(2)	Relates to the amount of data that will be held by user-defined communications support before sending a receive not ready (RNR) packet to the sending system. The recommended value for this field is 32, but any value between 1 and 128 may be used. **Note:** A receive ready (RR) packet will be sent when the user-defined communications application program receives some of the data.
reserved	CHAR(30)	Not used. This field should be set to hexadecimal zeros.

Data Unit Format - X.25 Operation X'B000' (Open a PVC Connection): The data for this operation starts at offset 0 from the top of the first data unit in the output buffer.

Table 2-17 shows the format of the data required for the X'B000' operation when opening a PVC connection

Table 2-17 (Page 1 of 3). Format of Data for X'B000' Operation (Open a PVC Connection)

Field	Type	Description
reserved	CHAR(1)	This field must be set to X'00'.
reserved	CHAR(1)	Not used. This field should be set to hexadecimal zeros (X'00').
logical channel identifier	CHAR(2)	Specifies the logical channel identifier of the PVC to open. Any PVC configured for this line that is eligible to be used by the network controller that the link is using may be specified and must be in the range of X'0001' - X'0FFF'. See "QOLQLIND" on page 2-57 for information on determining the PVCs configured for this line that are eligible to be used by the network controller the link is using.
transmit packet size	BINARY(2)	Specifies the requested transmit packet size for this connection. The valid values are 64, 128, 256, 512, and 1024. The value specified must be less than or equal to the transmit maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the transmit default packet size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the transmit maximum packet size and the transmit default packet size configured for this line.

Table 2-17 (Page 2 of 3). Format of Data for X'B000' Operation (Open a PVC Connection)

Field	Type	Description
transmit window size	BINARY(2)	Specifies the requested transmit window size for this connection. The valid values are as follows: **1 - 7** When modulus 8 is configured for this line. **1 - 15** When modulus 128 is configured for this line. **X'FFFF'** Use the transmit default window size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the modulus value and the transmit default window size configured for this line.
receive packet size	BINARY(2)	Specifies the requested receive packet size for this connection. The valid values are 64, 128, 256, 512, and 1024. The value specified must be less than or equal to the receive maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the receive default packet size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the receive maximum packet size and the receive default packet size configured for this line.
receive window size	BINARY(2)	Specifies the requested receive window size for this connection. The valid values are as follows: **1 - 7** When modulus 8 is configured for this line. **1 - 15** When modulus 128 is configured for this line. **X'FFFF'** Use the receive default window size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the modulus value and the receive default window size configured for this line.
reserved	CHAR(32)	Not used. This field should be set to hexadecimal zeros.
delivery confirmation support	CHAR(1)	Specifies the X.25 delivery confirmation bit (D-bit) support for this connection. The valid values are as follows: **X'00'** D-bit will be supported for sending data but not for receiving data. **Note:** When this value is selected and an X.25 packet is received with the D-bit set on, the input/output processor (IOP) will send a reset packet. **X'01'** D-bit will be supported for sending data and for receiving data.
reserved	CHAR(427)	This field must be set to hexadecimal zeros.

Table 2-17 (Page 3 of 3). Format of Data for X'B000' Operation (Open a PVC Connection)

Field	Type	Description
control information	CHAR(1)	Specifies control information for this connection. This is a bit-sensitive field with bit 0 (leftmost bit) defined for reset support. The remaining bits are undefined and should be set off ('0'B). The valid values for bit 0 are as follows: **'0'B** Resets are not supported on this connection. When this value is selected, the X'BF00' output operation will not be valid on this connection. Also, a reset indication packet received on this connection will cause the connection to be ended. **'1'B** Resets are supported on this connection. When this value is selected, the X'BF00' output operation will be valid on this connection. Also, the user-defined communications application program will be required to handle reset indications received on this connection. For example, consider the following values for the control information field: **X'00'** Resets are not supported on this connection. **X'80'** Resets are supported on this connection.
reserved	CHAR(3)	Not used. This field should be set to hexadecimal zeros.
maximum data unit assembly size	BINARY(4)	Specifies the maximum number of bytes of user data that can be received in a complete X.25 packet sequence on this connection. If this limit is exceeded, the connection will be ended. Any value between 1024 and 32767 may be used. **Notes:** 1. If the number of bytes of user data received in a complete X.25 packet sequence is more than can fit into one data unit of the input buffer, the more data indicator field in the corresponding element of the input buffer descriptor will be set to X'01' and the remaining user data will be filled in the next data unit. See "QOLRECV" on page 2-40 for more information. 2. There is no limit of the number of bytes of user data that can be sent in a complete X.25 packet sequence. However, the QOLSEND program may need to called more than once.
automatic flow control	BINARY(2)	Relates to the amount of data that will be held by user-defined communications support before sending a receive not ready (RNR) packet to the sending system. The recommended value for this field is 32, but any value between 1 and 128 may be used. **Note:** A receive ready (RR) packet will be sent when the user-defined communications application program receives some of the data.
reserved	CHAR(30)	Not used. This field should be set to hexadecimal zeros.

X.25 Operation X'B100'

This operation is used to either send a clear packet on an SVC, close an SVC connection that was cleared by the remote system, or to close a PVC connection. The user-defined communications application program must provide the data for this operation in the first data unit of the output buffer. The output buffer descriptor is not used.

The format of the data required for the X'B100' operation is the same whether or not it is used to send a clear packet on an SVC or to close a PVC connection. The format of the data required for the X'B100' operation should be set to hexadecimal zeros if it is used to close an SVC connection that was previously cleared by the remote system.

Notes:

1. The AS/400 system provides the confirmation of the clear indication, however, the local user-defined communications application must issue the X'B100' operation to free the PCEP for the connection.
2. Closing a PVC connection will cause a reset packet to be sent to the remote system.

Data Unit Format - X.25 Operation X'B100': The data for this operation starts at offset 0 from the top of the first data unit in the output buffer.

Table 2-18 shows the format of the data required for the X'B100' operation.

Table 2-18. Format of Data for X'B100' Operation

Field	**Type**	**Description**
reserved	CHAR(2)	Not used. This field should be set to hexadecimal zeros (X'0000').
cause code	CHAR(1)	Specifies the X.25 cause code.
diagnostic code	CHAR(1)	Specifies the X.25 diagnostic code.
reserved	CHAR(4)	Not used. This field should be set to hexadecimal zeros.
X.25 facilities length	BINARY(1)	Specifies the number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be used. This field is not used for PVC connections and should be set hexadecimal zeros (X'0000').
X.25 facilities	CHAR(109)	Specifies the X.25 facilities data. This field is not used for PVC connections and should be set hexadecimal zeros (X'0000').
reserved	CHAR(48)	Not used. This field should be set to hexadecimal zeros.
clear user data length	BINARY(2)	Specifies the number of bytes of data in the clear user data field. Any value between 0 and 128 may be used. This field is not used for PVC connections and should be set hexadecimal zeros (X'0000').
clear user data	CHAR(128)	Specifies the clear user data. **Note:** The CCITT standard recommends that this field only be present in conjunction with the fast select or call deflection selection facility. The AS/400 system does not enforce this restriction, however. This field is not used for PVC connections and should be set hexadecimal zeros (X'0000').
reserved	CHAR(216)	Not used. This field should be set to hexadecimal zeros.

X.25 Operation X′B400′

This operation is used to accept an incoming SVC call. The user-defined communications application program must provide the data for this operation in the first data unit of the output buffer. The output buffer descriptor is not used.

Note: Notification of incoming calls are received from the QOLRECV program with operation X′B201′. See "QOLRECV" on page 2-40 for more information.

Data Unit Format - X.25 Operation X′B400′: The data for this operation starts at offset 0 from the top of the first data unit in the output buffer.

Table 2-19 shows the format of the data required for the X′B400′ operation.

Table 2-19 (Page 1 of 3). Format of Data for X′B400′ Operation

Field	Type	Description
reserved	CHAR(1)	This field must be set to X′00′.
reserved	CHAR(3)	Not used. This field should be set to hexadecimal zeros (X′000000′).
transmit packet size	BINARY(2)	Specifies the transmit packet size for this connection. The valid values are 64, 128, 256, 512, and 1024. The value specified must be less than or equal to the transmit maximum packet size configured for this line. The special value of X′FFFF′ may be specified to use the transmit default packet size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the transmit maximum packet size and the transmit default packet size configured for this line.
transmit window size	BINARY(2)	Specifies the transmit window size for this connection. The valid values are as follows: **1 - 7** When modulus 8 is configured for this line. **1 - 15** When modulus 128 is configured for this line. **X′FFFF′** Use the transmit default window size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the modulus value and the transmit default window size configured for this line.
receive packet size	BINARY(2)	Specifies the receive packet size for this connection. The valid values are 64, 128, 256, 512, and 1024. The value specified must be less than or equal to the receive maximum packet size configured for this line. The special value of X′FFFF′ may be specified to use the receive default packet size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the receive maximum packet size and the receive default packet size configured for this line.
receive window size	BINARY(2)	Specifies the receive window size for this connection. The valid values are as follows: **1 - 7** When modulus 8 is configured for this line. **1 - 15** When modulus 128 is configured for this line. **X′FFFF′** Use the receive default window size configured for this line. See "QOLQLIND" on page 2-57 for information on determining the modulus value and the receive default window size configured for this line.
reserved	CHAR(32)	Not used. This field should be set to hexadecimal zeros.

Table 2-19 (Page 2 of 3). Format of Data for X'B400' Operation

Field	Type	Description
delivery confirmation support	CHAR(1)	Specifies if the X.25 delivery confirmation bit (D-bit) should be set on or off in the call accept packet. This also specifies the D-bit support for this connection. The valid values are as follows: **X'00'** Set the D-bit off in the call accept packet. D-bit will be supported for sending data but not for receiving data. **Note:** When this value is selected and an X.25 packet is received with the D-bit set on, the input/output processor (IOP) will send a reset packet. **X'01'** Set the D-bit on in the call accept packet. D-bit will be supported for sending data and for receiving data.
reserved	CHAR(11)	Not used. This field should be set to hexadecimal zeros.
X.25 facilities length	BINARY(1)	Specifies the number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be used. **Note:** The AS/400 system codes the packet and window size facilities in this field, if necessary. The total length of all facilities can not exceed 109 bytes.
X.25 facilities	CHAR(109)	Specifies the X.25 facilities data. **Note:** The application programmer should not code the facilities for packet or window sizes in this field. By doing so, this field could contain duplicate facilities, which may not be consistently supported by all X.25 networks.
reserved	CHAR(306)	Not used. This field should be set to hexadecimal zeros.
control information	CHAR(1)	Specifies control information for this connection. This is a bit-sensitive field with bit 0 (leftmost bit) defined for reset support. The remaining bits are undefined and should be set off ('0'B). The valid values for bit 0 are as follows: **'0'B** Resets are not supported on this connection. When this value is selected, the X'BF00' output operation will not be valid on this connection. Also, a reset indication packet received on this connection will cause the connection to be ended. **'1'B** Resets are supported on this connection. When this value is selected, the X'BF00' output operation will be valid on this connection. Also, the user-defined communications application program will be required to handle reset indications received on this connection. For example, consider the following values for the control information field: **X'00'** Resets are not supported on this connection. **X'80'** Resets are supported on this connection.
reserved	CHAR(3)	Not used. This field should be set to hexadecimal zeros.

Table 2-19 (Page 3 of 3). Format of Data for X'B400' Operation

Field	Type	Description
maximum data unit assembly size	BINARY(4)	Specifies the maximum number of bytes of user data that can be received in a complete X.25 packet sequence on this connection. If this limit is exceeded, the connection will be ended. Any value between 1024 and 32767 may be used. **Notes:** 1. If the number of bytes of user data received in a complete X.25 packet sequence is more than can fit into one data unit of the input buffer, the more data indicator field in the corresponding element of the input buffer descriptor will be set to X'01' and the remaining user data will be filled in the next data unit. See "QOLRECV" on page 2-40 for more information. 2. There is no limitation on the number of bytes of user data that can be sent in a complete X.25 packet sequence. However, the QOLSEND program may need to called more than once.
automatic flow control	BINARY(2)	Relates to the amount of data that will be held by user-defined communications support before sending a receive not ready (RNR) packet to the sending system. The recommended value for this field is 32, but any value between 1 and 128 may be used. **Note:** A receive ready (RR) packet will be sent when the user-defined communications application program receives some of the data.
reserved	CHAR(30)	Not used. This field should be set to hexadecimal zeros.

X.25 Operation X'BF00'

This operation is used to send a reset request packet or a reset confirmation packet on an X.25 SVC or PVC connection. The user-defined communications application program must provide the X.25 cause and diagnostic codes required for this operation in the first data unit of the output buffer. The output buffer descriptor is not used.

Information indicating whether a reset request or reset confirmation packet was sent will be returned when notification of the completion of the X'BF00' operation is received from the QOLRECV program (operation X'BF01'). This information will be in the diagnostic data parameter of the QOLRECV program. See "QOLRECV" on page 2-40 for more information.

A reset confirmation packet will be sent under the following conditions:

- after a reset indication packet has been received on the connection and the user-defined communications application program has received it from the QOLRECV program (X'B301' operation, 83/3202 return and reason code)
- after a reset indication packet has been received on the connection but before the user-defined communications application program has received it from the QOLRECV program
- when a reset indication packet is received on the connection at the same time the X'BF00' output operation is issued

  This is known as a reset collision. In this case, user-defined communications support will discard the reset indication and, therefore, the user-defined communications application program will not receive it from the QOLRECV program. However, the cause and diagnostic codes from the reset indication will be returned in the diagnostic data parameter of the QOLRECV program when notification of the completion of the X'BF00' operation is received. See "QOLRECV" on page 2-40 for more information.

A reset request packet will be sent when none of the above conditions are true.

Notes:

1. Data not yet received by the user-defined communications application program on a connection will *not* be deleted when a X'BF00' operation is issued on that connection. This data will be received before the notification of the completion of the X'BF00' operation is received from the QOLRECV program (operation X'BF01'). Data received after the notification of the completion of the X'BF00' operation is received should be treated as new data.
2. The X'BF00' operation is only valid on connections that support resets. See "X.25 Operation X'B000'" on page 2-25 and "X.25 Operation X'B400'" on page 2-32 for more information on specifying reset support.

Data Unit Format - X.25 Operation X'BF00': Just the first data unit in the output buffer is used for this operation. The first byte (offset 0 from the top of the first data unit) contains the X.25 cause code. The second byte (offset 1 from the top of the first data unit) contains the X.25 diagnostic code.

Return and Reason Codes

The return and reason codes that can be returned from the QOLSEND program depend on the type of communications line the link is using and on the operation that was requested.

LAN Return and Reason Codes

Return and Reason Codes for LAN Operation X'0000'

Table 2-20 (Page 1 of 2). Return and Reason Codes for LAN Operation X'0000'

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
80/2200	Data queue error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Then correct the error, enable the link, and try the request again.
80/2401	Output buffer or output buffer descriptor error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Then correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the user-defined communications application program by escape message CPF91F0 or CPF91F1. However, the user-defined communications application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the user-defined communications application program, then report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.

Table 2-20 (Page 2 of 2). Return and Reason Codes for LAN Operation X'0000'

Return / Reason Code	Meaning	Recovery
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Correct the condition, enable the link, and try the request again.
80/8000	The amount of user data in a data unit of the output buffer is greater than the maximum frame size allowed on the communications line the link is using. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled. Correct the error, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the user-defined communications application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/1006	Output operation not valid.	Correct the operation parameter. Try the request again.
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/1008	Number of data units not valid.	Correct the number of data units parameter. Try the request again.
83/1998	The amount of data in a data unit of the output buffer is not correct.	Correct the amount of user data, or the total amount of logical link control (LLC) information, routing information, and user data in the offending data unit. Try the request again.
83/1999	Incorrect data in a data unit of the output buffer. The error offset field in the diagnostic data parameter will point to the incorrect data.	Correct the incorrect data. Try the request again.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue. If the link was successfully enabled, try the request again.
83/4001	Link failure, system starting error recovery for this link.	Wait for the link to recover. Try the request again.
83/4003	Error detected by the input/output processor (IOP). The diagnostic data parameter will contain more information on this error.	Correct the error, and try the request again.

X.25 Return and Reason Codes

General X.25 Return and Reason Codes: Table 2-21 on page 2-37 shows the return and reason codes that can be received from the QOLSEND program for any requested operation.

Table 2-21. Return and Reason Codes Valid for All X.25 Operations

Return / Reason Code	Meaning	Recovery
80/2200	Data queue error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/2401	Output buffer or output buffer descriptor error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the user-defined communications application program by escape message CPF91F0 or CPF91F1. However, the user-defined communications application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the user-defined communications application program, report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Correct the condition, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the user-defined communications application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/1006	Output operation not valid.	Correct the operation parameter. Try the request again.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue. If the link was successfully enabled, try the request again.
83/3200	All resources are currently in use by asynchronous operations that have not yet completed.	Wait for at least one of the asynchronous operations to complete. Notification of completion of these operations will be received from the QOLRECV program. Try the request again.

Return and Reason Codes for X.25 Operation X'0000'

Table 2-22 (Page 1 of 2). Return and Reason Codes for X.25 Operation X'0000'

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/1008	Number of data units not valid.	Correct the number of data units parameter. Try the request again.

Table 2-22 (Page 2 of 2). Return and Reason Codes for X.25 Operation X'0000'

Return / Reason Code	Meaning	Recovery
83/1997	The amount of user data in a data unit of the output buffer is not a multiple of the negotiated transmit packet size, and the more data indicator in the corresponding element of the output buffer descriptor is set to X'01'.	Correct the amount of user data in the offending data unit. Try the request again.
83/1998	The amount of user data in a data unit of the output buffer is not correct.	Correct the amount of user data in the offending data unit. Try the request again.
83/3201	The maximum amount of incoming user data that can be held by user-defined communications support for the user-defined communications application program on this connection has been exceeded.	Wait to receive a failure notification from the QOLRECV program indicating this condition (X'B301' operation, 83/3201 return and reason code). Issue the X'B100' output operation to end the connection.
83/3202	A reset indication has been received on this connection. The X.25 cause and diagnostic code fields in the diagnostic data parameter will contain the cause and diagnostic codes of the reset indication.	Wait to receive notification from the QOLRECV program indicating this condition (X'B301' operation, 83/3202 return and reason code). Issue the X'BF00' output operation to send a reset confirmation packet.
83/3205	Connection not in a valid state.	Ensure the connection is in a valid state for this operation. Try the request again.
83/4001	Link failure, system starting error recovery for this link.	Wait to receive a failure notification from the QOLRECV program indicating this condition (X'B301' operation, 83/4001 return and reason code). Issue the X'B100' output operation to end the connection.
83/4002	Connection failure.	Wait to receive a failure notification from the QOLRECV program indicating this condition (X'B301' operation, 83/4002 return and reason code). Issue the X'B100' output operation to end the connection.

Return and Reason Codes for X.25 Operation X'B000'

Table 2-23. Return and Reason Codes for X.25 Operation X'B000'

Return / Reason Code	Meaning	Recovery
0/0	Operation initiated.	Wait for notification of the completion of the X'B000' operation from the QOLRECV program (X'B001' operation).
83/4005	All connections are currently in use.	Wait for a connection to become available and try the request again.

Return and Reason Codes for X.25 Operation X'B100'

Table 2-24. Return and Reason Codes for X.25 Operation X'B100'

Return / Reason Code	Meaning	Recovery
0/0	Operation initiated.	Wait for notification of the completion of the X'B100' operation from the QOLRECV program (X'B101' operation).
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/3205	Connection not in a valid state.	Ensure the connection is in a valid state for this operation. Try the request again.

Return and Reason Codes for X.25 Operation X'B400'

Table 2-25. Return and Reason Codes for X.25 Operation X'B400'

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/1999	Incorrect data in a data unit of the output buffer. The error offset field in the diagnostic data parameter will point to the incorrect data.	Correct the incorrect data. Try the request again.
83/3205	Connection not in a valid state.	Ensure the connection is in a valid state for this operation. Try the request again.
83/4001	Link failure, system starting error recovery for this link.	Issue the X'B100' output operation to end the connection.
83/4004	Inbound call timed out.	Issue the X'B100' output operation to end the connection.

Return and Reason Codes for X.25 Operation X'BF00'

Table 2-26. Return and Reason Codes for X.25 Operation X'BF00'

Return / Reason Code	Meaning	Recovery
0/0	Operation initiated.	Wait for notification of the completion of the X'BF00' operation from the QOLRECV program (X'BF01' operation).
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/3205	Connection not in a valid state.	Ensure the connection is in a valid state for this operation. Try the request again.

QOLRECV

```
CALL QOLRECV(return code,
             reason code,
             existing user connection end point ID,
             new provider connection end point ID,
             operation,
             number of data units,
             data available,
             diagnostic data,
             communications handle)
```

Parameter List

Table 2-27 (Page 1 of 3). QOLRECV Parameter List

Parameter	Use	Type	Description
return code	output	BINARY(4)	Specifies the recovery action to take. See "Return and Reason Codes" on page 2-52.
reason code	output	BINARY(4)	Specifies the error that occurred. See "Return and Reason Codes" on page 2-52.
existing user connection end point ID	output	BINARY(4)	Specifies the user connection end point (UCEP) ID that the data was received on. For links using a token-ring or Ethernet communications line, the content of this parameter will always be 1. For links using an X.25 communications line, the content of this parameter is only valid when the operation parameter is X'0001', X'B001', X'B101', X'B301', or X'BF01'. It will contain the UCEP ID that was provided in the new user connection end point ID parameter on the call to the QOLSEND program with operation X'B000' or X'B400'. **Note:** If an incoming X.25 SVC call is rejected by the user-defined communications application program by calling the QOLSEND program with operation X'B100', the content of this parameter will be set to zero when notification of the completion of the X'B100' operation is received from the QOLRECV program (operation X'B101').
new provider connection end point ID	output	BINARY(4)	Specifies the provider connection end point (PCEP) ID for the connection that is to be established. This identifier must be used on all subsequent calls to the QOLSEND program for this connection. The content of this parameter is only valid for links using an X.25 communications line and when the operation parameter is X'B201'.

Table 2-27 (Page 2 of 3). QOLRECV Parameter List

Parameter	Use	Type	Description
operation	output	CHAR(2)	Specifies the type of data received by the user-defined communications application program. The valid values are as follows: **X'0001'** User data. **X'B001'** Completion of the X'B000' output operation. This is only valid for links using an X.25 communications line. **X'B101'** Completion of the X'B100' output operation. This is only valid for links using an X.25 communications line. **X'B201'** Incoming X.25 switched virtual circuit (SVC) call. This is only valid for links using an X.25 communications line. **X'B301'** Connection failure or reset indication received. This is only valid for links using an X.25 communications line. **X'BF01'** Completion of the X'BF00' output operation. This is only valid for links using an X.25 communications line. **Note:** The special value of X'0000' will be returned in the operation parameter to indicate no data was received from the QOLRECV program. See "Return and Reason Codes" on page 2-52 for more information.
number of data units	output	BINARY(4)	Specifies the number of data units in the input buffer that contain data. Any value between 1 and the number of data units created in the input buffer may be returned when the operation parameter is X'0001'. Otherwise, any value between 0 and 1 may be returned. **Note:** The number of data units created in the input buffer was returned in the data units created parameter on the call to the QOLELINK program. See "QOLELINK" on page 2-3 for more information.
data available	output	CHAR(1)	Specifies if more data is available for the user-defined communications application program to receive. The valid values are as follows: **X'00'** No more data is available for the user-defined communications application program to receive. **X'01'** More data is available for the user-defined communications application program to receive. The QOLRECV program should be called again. **Note:** An incoming-data entry will be sent to the data queue only when the content of this parameter is X'00' and then more data is subsequently available to be received. See "Incoming-Data Entry" on page 6-4 for more information.
diagnostic data	output	CHAR(40)	Specifies additional diagnostic data. See "Format of Diagnostic Data Parameter" on page 2-42 for more information. The content of this parameter is only valid when the operation parameter is X'B001', X'B101', X'B301', or X'BF01'.

Table 2-27 (Page 3 of 3). QOLRECV Parameter List

Parameter	Use	Type	Description
communications handle	input	CHAR(10)	Specifies the name of the link on which to receive the data.

Format of Diagnostic Data Parameter

The format of the diagnostic data parameter is shown below. The contents of the fields within this parameter are only valid on X′B001′, X′B101′, X′B301′, and X′BF01′ operations for the indicated return and reason codes.

Table 2-28 (Page 1 of 2). Diagnostic Data Parameter

Field	Type	Description
reserved	CHAR(2)	Not used.
error code	CHAR(4)	Specifies hexadecimal diagnostic information that can be used to determine recovery actions. See “Error Codes” on page 5-11 for more information. The content of this field is only valid for 83/4001 and 83/4002 return/reason codes.
time stamp	CHAR(8)	Specifies the time the error occurred. The content of this field is only valid for 83/4001 and 83/4002 return/reason codes.
error log identifier	CHAR(4)	Specifies the hexadecimal identifier that can be used for locating error information in the error log. The content of this field is only valid for 83/4001 and 83/4002 return/reason codes.
reserved	CHAR(10)	Not used.

Table 2-28 (Page 2 of 2). Diagnostic Data Parameter

Field	**Type**	**Description**
indicators	CHAR(1)	Specifies the indicators that the user-defined communications application program can use to diagnose a potential error condition. This is a bit-sensitive field. The valid values for bit 0 (leftmost bit) are as follows: **'0'B** Either there is no message in the QSYSOPR message queue, or there is a message and it does not have the capability to run problem analysis report (PAR) to determine the cause of the error. **'1'B** There is a message in the QSYSOPR message queue for this error, and it does have the capability to run problem analysis report (PAR) to determine the cause of the error. The valid values for bit 1 are as follows: **'0'B** The line error can be retried. **'1'B** The line error is not able to be restarted. The valid values for bit 2 are as follows: **'0'B** The cause and diagnostic codes fields are not valid. **'1'B** The cause and diagnostic codes fields are valid. The valid values for bit 3 are as follows: **'0'B** The error has not been reported to the system operator message queue. **'1'B** The error has been reported to the system operator message queue. The valid values for bit 4 are as follows: **'0'B** A reset request packet was transmitted on the network **'1'B** A reset confirmation packet was transmitted on the network instead of a reset request packet. The content of this field is only valid for operation X'BF01' with 00/0000 return/reason codes. The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes, and 00/0000 return/reason codes for operation X'BF01'.
X.25 cause code	CHAR(1)	Specifies additional information on the condition reported. See the *X.25 Network Guide* for interpreting the values of this field. The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes.
X.25 diagnostic code	CHAR(1)	Specifies additional information on the condition reported. See the *X.25 Network Guide* for interpreting the values of this field. The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes.
reserved	CHAR(1)	Not used.
error offset	BINARY(4)	Specifies the offset from the top of the input buffer to the incorrect data in the input buffer. The content of this field is only valid for a 83/1999 return/reason code.
reserved	CHAR(4)	Not used.

Description of Function

The QOLRECV program is called by a user-defined communications application program to perform input on a link that is currently enabled in the job in which the user-defined communications application program is running. The type of data received is returned in the operation parameter. The data will be returned in the input buffer that was created when the link was enabled. For X′0001′ operations, a description of that data will also be returned in the input buffer descriptor that was created when the link was enabled.

The types of data that can be received from the QOLRECV program depend on the type of communications line the link is using. See "LAN Input Operations" for more information on the types of data that can be received on links using a token-ring or Ethernet communications line. See "X.25 SVC and PVC Input Operations" on page 2-46 for more information on the types of data that can be received on links using an X.25 communications line.

Note: The QOLRECV program should only be called when the user-defined communications support has data available to be received. This is indicated either by an incoming-data entry on the data queue or by the data available parameter on the QOLRECV program.

LAN Input Operations

The only type of data that can be received from the QOLRECV program on links using a token-ring or Ethernet communications line is user data (operation X′0001′). User-defined communications support will return the following information for each data frame received from the QOLRECV program:

- logical link control (LLC) information, optional routing information, and user data in the next data unit of the input buffer, starting with the first data unit
- a description, in the corresponding element of the input buffer descriptor, of the information in that data unit

For example, suppose two data frames came in from the network and the user-defined communications application program was notified of this by an incoming-data entry on the data queue. On return from the QOLRECV program, the information for the first frame would be in the first data unit of the input buffer and described in the first element of the input buffer descriptor. The information for the second frame would be in the second data unit of the input buffer and described in the second element of the input buffer descriptor. The number of data units parameter would be set to 2.

Data Unit Format - LAN Operation X′0001′: Each data frame received from the QOLRECV program corresponds to a data unit in the input buffer. The information in each of these data units is made up of LLC information, optional routing data, and user data.

The LLC information starts at offset 0 from the top of the data unit. The routing information (if any) starts immediately after the LLC information, and the user data starts immediately after the routing information. If there isn't any routing information, the user data starts immediately after the LLC information.
Table 2-29 on page 2-45 shows the format of the LLC information.

Table 2-29. Format of the LLC Information

Field	Type	Description
length of LLC information	BINARY(2)	Specifies the length of the LLC information in the data unit. This will always be set to 16.
sending adapter address	CHAR(6)	Specifies, in packed form, the adapter address from which this frame was sent. The possible values returned in this field depend on the filters activated for this link. See "QOLSETF" on page 2-10 for more information. **Note:** Because user-defined communications support only allows connectionless service over LANs, all frames received on a single call to the QOLRECV program may not have the same source adapter address.
DSAP address	CHAR(1)	Specifies the service access point on which the AS/400 system received this frame. The possible values returned in this field depend on the filters activated for this link. See "QOLSETF" on page 2-10 for more information. **Note:** The Ethernet Version 2 standard does not define a DSAP address in an Ethernet Version 2 frame. Therefore, when receiving Ethernet Version 2 frames, the DSAP address will be null (X'00').
SSAP address	CHAR(1)	Specifies the service access point on which the source system sent this frame. The possible values returned in this field depend on the filters activated for this link. See "QOLSETF" on page 2-10 for more information. **Note:** The Ethernet Version 2 standard does not define a SSAP address in an Ethernet Version 2 frame. Therefore, when receiving Ethernet Version 2 frames, the SSAP address will be null (X'00').
reserved	CHAR(2)	Not used.
length of routing information	BINARY(2)	Specifies the length of the routing information in the data unit. For links using a token-ring communications line, any value between 0 and 18 may be returned, where 0 indicates that there is no routing information. For links using an Ethernet communications line, the content of this field is not applicable and will be set to 0 indicating that there is no routing information.
length of user data	BINARY(2)	Specifies the length of the user data in the data unit. This will be less than or equal to the maximum frame size allowed on the service access point returned in the DSAP address field. See "QOLQLIND" on page 2-57 to determine the maximum frame size allowed on the service access point returned in the DSAP address field. For Ethernet Version 2 frames, this will be at least 48 and not more than 1502 (including 2 bytes for the Ethernet type field). **Note:** Ethernet 802.3 frames will be padded when the user data is less than 48 bytes.

Input Buffer Descriptor Element Format - LAN Operation X'0001': The information returned in each data unit of the input buffer will be described in the corresponding element of the input buffer descriptor.

Table 2-30 on page 2-46 shows the format of each element in the input buffer descriptor.

Table 2-30. Format of an Element in the Input Buffer Descriptor

Field	Type	Description
length	BINARY(2)	Specifies the number of bytes of information in the corresponding data unit of the input buffer. This will be equal to the length of the LLC information plus the length of the routing information plus the length of the user data specified in Table 2-29.
reserved	CHAR(30)	Not used.

X.25 SVC and PVC Input Operations

Table 2-31 shows the types of data that can be received from the QOLRECV program on links using an X.25 communications line.

Table 2-31. X.25 SVC and PVC Input Operations

Operation	Meaning
X'0001'	User data (SVC or PVC).
X'B001'	Completion of the X'B000' output operation (SVC or PVC).
X'B101'	Completion of the X'B100' output operation (SVC or PVC).
X'B201'	Incoming X.25 call (SVC).
X'B301'	Connection failure or reset indication (SVC or PVC).
X'BF01'	Completion of the X'BF00' output operation (SVC or PVC).

X.25 Operation X'0001'

This operation indicates that user data was received on an X.25 SVC or PVC connection. User-defined communications support will return the following information:

- user data in the next data unit of the input buffer, starting with the first data unit
- a description, in the corresponding element of the input buffer descriptor, of the user data in that data unit

For example, suppose two data units of user data came in from the network and the user-defined communications application program was notified of this by an incoming-data entry on the data queue. On return from the QOLRECV program, the first portion of the user data would be in the first data unit of the input buffer and described in the first element of the input buffer descriptor. The second portion of the user data would be in the second data unit of the input buffer and described in the second element of the input buffer descriptor. The number of data units parameter would be set to 2.

User-defined communications support will automatically reassemble the X.25 data packet(s) from a complete packet sequence into the next data unit of the input buffer. If the amount of user data in a complete packet sequence is more than what can fit into a data unit, the more data indicator field in the corresponding element of the input buffer descriptor will be set to X'01' and the next data unit will be used for the remaining user data, and so on.

Data Unit Format - X.25 Operation X'0001': Each data unit in the input buffer consists solely of user data and starts offset 0 from the top of the data unit.

Input Buffer Descriptor Element Format - X.25 Operation X'0001': The user data returned in each data unit of the input buffer will be described in the corresponding element of the input buffer descriptor.

Table 2-32 shows the format of each element in the input buffer descriptor.

Table 2-32. Format of an Element in the Input Buffer Descriptor

Field	**Type**	**Description**
length	BINARY(2)	Specifies the number of bytes of user data in the corresponding data unit of the input buffer. This will always be less than or equal to the X.25 user data size parameter that was specified on the call to the QOLELINK program when the link was enabled. See "QOLELINK" on page 2-3 for more information. **Note:** The maximum amount of user data in a data unit of the input buffer may be further limited by the maximum data unit assembly size for a connection. See "QOLSEND" on page 2-17 for more information.
more data indicator	CHAR(1)	Specifies if the remaining amount of user data from a complete X.25 packet sequence is more than can fit into the corresponding data unit. The valid values are as follows: **X'00'** The remaining amount of user data from a complete X.25 packet sequence fit into the corresponding data unit. **X'01'** The remaining amount of user data from a complete X.25 packet sequence could not all fit into the corresponding data unit. The next data unit will be used.
qualified data indicator	CHAR(1)	Specifies if the X.25 qualifier bit (Q-bit) was set on or off in all X.25 packets reassembled into the corresponding data unit. The valid values are as follows: **X'00'** The Q-bit was set off in all X.25 packets reassembled into the corresponding data unit. **X'01'** The Q-bit was set on in all X.25 packets reassembled into the corresponding data unit.
interrupt packet indicator	CHAR(1)	Specifies if the user data in the corresponding data unit was received in an X.25 interrupt packet. The valid values are as follows: **X'00'** The user data in the corresponding data unit was received in one or more data packets. **X'01'** The user data in the corresponding data unit was received in an X.25 interrupt packet.
delivery confirmation indicator	CHAR(1)	Specifies if the X.25 delivery confirmation bit (D-bit) was set on or off in all X.25 packets reassembled into the corresponding data unit. The valid values are as follows: **X'00'** The D-bit was set off in all X.25 packets reassembled into the corresponding data unit. **X'01'** The D-bit was set on in all X.25 packets reassembled into the corresponding data unit. **Note:** A packet-level confirmation is sent by the input/output processor (IOP) when a packet is received with the X.25 D-bit set on.
reserved	CHAR(26)	Not used.

X.25 Operation X'B001'

This operation indicates that a X'B000' output operation has completed. User-defined communications support will return the data for this operation (if any) in the first data unit of the input buffer. The input buffer descriptor is not used.

Data will be returned in the input buffer for the following return and reason codes:

- 0/0
- 83/1999
- 83/4002 (only when the number of data units parameter is set to one)

The format of the data returned in the input buffer for the X'B001' operation depends on whether the X'B000' output operation was used to initiate an SVC call or to open a PVC connection. Each format will be explained below.

Note: The formats below only apply to 0/0 and 83/4002 return and reason codes. When the X'B001' operation is received with a 83/1999 return and reason code, the data returned starts at offset 0 from the top of the first data unit in the input buffer and contains the data specified in the output buffer on the X'B000' output operation. See "QOLSEND" on page 2-17 for more information.

Data Unit Format - X.25 Operation X'B001' (Completion of SVC Call): The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Table 2-33 shows the format of the data returned for the X'B001' operation.

Table 2-33 (Page 1 of 2). Format of Data for X'B001' Operation (Completion of SVC Call)

Field	Type	Description
reserved	CHAR(2)	Not used.
logical channel identifier	CHAR(2)	Specifies the logical channel identifier assigned to the SVC connection. **Note:** The content of this field is only valid for a 0/0 return and reason code.
transmit packet size	BINARY(2)	Specifies the negotiated transmit packet size for this connection. **Note:** The content of this field is only valid for a 0/0 return and reason code.
transmit window size	BINARY(2)	Specifies the negotiated transmit window size for this connection. **Note:** The content of this field is only valid for a 0/0 return and reason code.
receive packet size	BINARY(2)	Specifies the negotiated receive packet size for this connection. **Note:** The content of this field is only valid for a 0/0 return and reason code.
receive window size	BINARY(2)	Specifies the negotiated receive window size for this connection. **Note:** The content of this field is only valid for a 0/0 return and reason code.
reserved	CHAR(32)	Not used.

Table 2-33 (Page 2 of 2). Format of Data for X'B001' Operation (Completion of SVC Call)

Field	Type	Description
delivery confirmation support	CHAR(1)	Specifies if the X.25 delivery confirmation bit (D-bit) was set on or off in the call connected packet. This also specifies the D-bit support for this connection. The valid values are as follows: **X'00'** The D-bit was set off in the call connected packet. D-bit will be supported for sending data but not for receiving data. **Note:** When this value is returned and an X.25 packet is received with the D-bit set on, the input/output processor (IOP) will send a reset packet. **X'01'** The D-bit was set on in the call connected packet. D-bit will be supported for sending data and for receiving data. **Note:** The content of this field is only valid for a 0/0 return and reason code.
reserved	CHAR(11)	Not used.
X.25 facilities length	BINARY(1)	Specifies the number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be returned.
X.25 facilities	CHAR(109)	Specifies the X.25 facilities data.
reserved	CHAR(48)	Not used.
call/clear user data length	BINARY(2)	Specifies the number of bytes of data in the call/clear user data field. Any value between 0 and 128 may be returned.
call/clear user data	CHAR(128)	For a 0/0 return and reason code, this specifies the call user data. For an 83/4002 return and reason code, this specifies the clear user data.
reserved	CHAR(168)	Not used.

Data Unit Format - X.25 Operation X'B001' (Completion of Open PVC): The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Table 2-34 shows the format of the data returned for the X'B001' operation.

Table 2-34. Format of Data for X'B001' Operation (Completion of Open PVC)

Field	Type	Description
reserved	CHAR(4)	Not used.
transmit packet size	BINARY(2)	Specifies the negotiated transmit packet size for this connection. **Note:** This will be the same as the requested transmit packet size specified on the X'B000' output operation.
transmit window size	BINARY(2)	Specifies the negotiated transmit window size for this connection. **Note:** This will be the same as the requested transmit window size specified on the X'B000' output operation.
receive packet size	BINARY(2)	Specifies the negotiated receive packet size for this connection. **Note:** This will be the same as the requested receive packet size specified on the X'B000' output operation.
receive window size	BINARY(2)	Specifies the negotiated receive window size for this connection. **Note:** This will be the same as the requested receive window size specified on the X'B000' output operation.
reserved	CHAR(500)	Not used.

X.25 Operation X′B101′

This operation indicates that a X′B100′ output operation has completed. User-defined communications support will return the data for this operation (if any) in the first data unit of the input buffer. The input buffer descriptor is not used.

Data will be returned in the input buffer for the following return and reason codes:

- 0/0 (only when the number of data units parameter is set to one)
- 83/1999

Note: The format below only applies for a 0/0 return and reason code. When the X′B101′ operation is received with an 83/1999 return and reason code, the data returned starts at offset 0 from the top of the first data unit in the input buffer and contains the data specified in the output buffer on the X′B100′ output operation. See "QOLSEND" on page 2-17 for more information.

Data Unit Format - X.25 Operation X′B101′: The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Table 2-35 shows the format of the data returned for the X′B101′ operation.

Table 2-35. Format of Data for X′B101′ Operation

Field	Type	Description
clear type	CHAR(2)	Specifies the type of clear user data returned. The valid values are as follows: **X′0001′** Clear confirmation data included. **X′0002′** Clear indication data included.
cause code	CHAR(1)	Specifies the X.25 cause code.
diagnostic code	CHAR(1)	Specifies the X.25 diagnostic code.
reserved	CHAR(4)	Not used.
X.25 facilities length	BINARY(1)	Specifies the number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be returned.
X.25 facilities	CHAR(109)	Specifies the X.25 facilities data.
reserved	CHAR(48)	Not used.
clear user data length	BINARY(2)	Specifies the number of bytes of data in the clear user data field. Any value between 0 and 128 may be returned.
clear user data	CHAR(128)	Specifies the clear user data.
reserved	CHAR(216)	Not used.

X.25 Operation X′B201′

This operation indicates that an incoming X.25 SVC call was received. User-defined communications support will return the data for this operation in the first data unit of the input buffer. The input buffer descriptor is not used.

Note: It is the responsibility of the user-defined communications application program to either accept or reject the incoming call. This is done by calling the QOLSEND program with operation X′B400′ or X′B100′, respectively.

Data Unit Format - X.25 Operation X'B201': The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Table 2-36 shows the format of the data returned for the X'B201' operation.

Table 2-36. Format of Data for X'B201' Operation

Field	**Type**	**Description**
reserved	CHAR(2)	Not used.
logical channel identifier	CHAR(2)	Specifies the logical channel identifier assigned to the incoming SVC call.
transmit packet size	BINARY(2)	Specifies the requested transmit packet size for this connection.
transmit window size	BINARY(2)	Specifies the requested transmit window size for this connection.
receive packet size	BINARY(2)	Specifies the requested receive packet size for this connection.
receive window size	BINARY(2)	Specifies the requested receive window size for this connection.
reserved	CHAR(7)	Not used.
Calling DTE address length	BINARY(1)	Specifies the number of binary coded decimal (BCD) digits in the calling DTE address.
Calling DTE address	CHAR(16)	Specifies, in binary coded decimal (BCD), the calling DTE address. The address will be left justified and padded on the right with BCD zeros.
reserved	CHAR(8)	Not used.
delivery confir-mation support	CHAR(1)	Specifies if the X.25 delivery confirmation bit (D-bit) was set on or off in the incoming call packet. The valid values are as follows: **X'00'** The D-bit was set off in the incoming call packet. **X'01'** The D-bit was set on in the incoming call packet.
reserved	CHAR(9)	Not used.
reverse charging indicator	CHAR(1)	Specifies reverse charging options. The valid values are as follows: **X'00'** Reverse charging not requested. **X'01'** Reverse charging requested.
fast select indi-cator	CHAR(1)	Specifies fast select options. The valid values are as follows: **X'00'** Fast select not requested. **X'01'** Fast select with restriction requested. **X'02'** Fast select without restriction requested.
X.25 facilities length	BINARY(1)	Specifies the number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be returned.
X.25 facilities	CHAR(109)	Specifies the X.25 facilities data.
reserved	CHAR(48)	Not used.
call user data length	BINARY(2)	Specifies the number of bytes of data in the call user data field. Any value between 0 and 128 may be returned.
call user data	CHAR(128)	Specifies the call user data. **Note:** The AS/400 system treats the first byte of call user data as the protocol identifier (PID).
reserved	CHAR(168)	Not used.

X.25 Operation X'B301'

This operation indicates that a failure has occurred, or a reset indication has been received, on an X.25 SVC or PVC connection. User-defined communications support will return data for this operation in the first data unit of the input buffer only on a 83/4002 return and reason code when the number of data units parameter is set to one. The input buffer descriptor is not used.

Note: The diagnostic data parameter will contain the X.25 cause and diagnostic codes when a reset indication is received.

Data Unit Format - X.25 Operation X'B301': The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Table 2-37 shows the format of the data returned for the X'B301' operation.

Table 2-37. Format of Data for X'B301' Operation

Field	Type	Description
reserved	CHAR(8)	Not used.
X.25 facilities length	BINARY(1)	Specifies the number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be returned.
X.25 facilities	CHAR(109)	Specifies the X.25 facilities data.
reserved	CHAR(48)	Not used.
clear user data length	BINARY(2)	Specifies the number of bytes of data in the clear user data field. Any value between 0 and 128 may be returned.
clear user data	CHAR(128)	Specifies the clear user data.
reserved	CHAR(216)	Not used.

X.25 Operation X'BF01'

This operation indicates that a X'BF00' output operation has been completed. Neither the input buffer nor the input buffer descriptor is used for this operation.

Note: When the X'BF01' operation is received with a 0/0 return and reason code, the diagnostic data parameter will contain information indicating if a reset request or reset confirmation packet was sent.

Return and Reason Codes

The return and reason codes that can be returned from the QOLRECV program depend on the type of communications line the link is using and on the type of data (operation) that was received.

LAN Return and Reason Codes

Return and Reason Codes Indicating No Data Received: Table 2-38 on page 2-53 shows the return and reason codes that indicate data could not be received from the QOLRECV program.

Note: When these return and reason codes are returned, all output parameters except the return and reason codes will contain hexadecimal zeros.

Table 2-38. Return and Reason Codes Indicating No Data Received

Return / Reason Code	**Meaning**	**Recovery**
0/3203	No data available to be received.	Ensure that user-defined communications support has data available to be received before calling the QOLRECV program. Try the request again.
80/2200	Data queue error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/2401	Input buffer or input buffer descriptor error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the user-defined communications application program by escape message CPF91F0 or CPF91F1. However, the user-defined communications application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the user-defined communications application program, then report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Correct the condition, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the user-defined communications application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue. If the link was successfully enabled, try the request again.

Return and Reason Codes for LAN Operation X'0001'

Table 2-39. Return and Reason Codes for LAN Operation X'0001'

Return / Reason Code	**Meaning**	**Recovery**
0/0	User data received successfully.	Continue processing.

X.25 Return and Reason Codes

Return and Reason Codes Indicating No Data Received: Table 2-40 shows the return and reason codes that indicate data could not be received from the QOLRECV program.

Note: When these return and reason codes are returned, all output parameters except the return and reason codes will contain hexadecimal zeros.

Table 2-40. Return and Reason Codes Indicating No Data Received

Return / Reason Code	Meaning	Recovery
0/3203	No data available to be received.	Ensure that user-defined communications support has data available to be received before calling the QOLRECV program. Try the request again.
80/2200	Data queue error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/2401	Input buffer or input buffer descriptor error detected. Escape message CPF91F1 will be sent to the user-defined communications application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the user-defined communications application program by escape message CPF91F0 or CPF91F1. However, the user-defined communications application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the user-defined communications application program, then report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Correct the condition, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the user-defined communications application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue. If the link was successfully enabled, try the request again.

Return and Reason Codes for X.25 Operation X'0001'

Table 2-41. Return and Reason Codes for X.25 Operation X'0001'

Return / Reason Code	Meaning	Recovery
0/0	User data received successfully.	Continue processing.

Return and Reason Codes for X.25 Operation X'B001'

Table 2-42. Return and Reason Codes for X.25 Operation X'B001'

Return / Reason Code	Meaning	Recovery
0/0	The X'B000' output operation was successful.	Continue processing.
83/1999	Incorrect data was specified in output buffer when the X'B000' output operation was issued. **Note:** The data specified in the output buffer will be copied into the input buffer and the error offset field in the diagnostic data parameter will point to the incorrect data.	Correct the incorrect data. Then, try the X'B000' output operation again.
83/3204	Connection ending because a X'B100' output operation was issued.	Wait for notification of the completion of the X'B100' output operation from the QOLRECV program (X'B101' operation).
83/4001	Link failure, system starting error recovery for this link. The connection has ended.	Wait for the link to recover. Then, try the X'B000' output operation again.
83/4002	Connection failure. The connection has ended. The diagnostic data parameter will contain more information on this error.	Correct any errors and try the X'B000' output operation again.
83/4005	All SVC channels are currently in use, or the requested PVC channel is already in use.	Wait for a virtual circuit to become available. Then, try the X'B000' output operation again.

Return and Reason Codes for X.25 Operation X'B101'

Table 2-43. Return and Reason Codes for X.25 Operation X'B101'

Return / Reason Code	Meaning	Recovery
0/0	The X'B100' output operation was successful. The connection has ended.	Continue processing.
83/1007	Connection identifier not valid because connection has already ended.	Continue processing.
83/1999	Incorrect data was specified in output buffer when the X'B100' output operation was issued. **Note:** The data specified in the output buffer will be copied into the input buffer and the error offset field in the diagnostic data parameter will point to the incorrect data.	Correct the incorrect data. Then, try the X'B100' output operation again.

Return and Reason Codes for X.25 Operation X'B201'

Table 2-44. Return and Reason Codes for X.25 Operation X'B201'

Return / Reason Code	Meaning	Recovery
0/0	Incoming X.25 SVC call received successfully.	Continue processing.

Return and Reason Codes for X.25 Operation X'B301'

Table 2-45. Return and Reason Codes for X.25 Operation X'B301'

Return / Reason Code	Meaning	Recovery
83/3201	The maximum amount of incoming user data that can be held by user-defined communications support for the user-defined communications application program on this connection has been exceeded.	Issue the X'B100' output operation to end the connection.
83/3202	A reset indication has been received on this connection. The X.25 cause and diagnostic code fields in the diagnostic data parameter will contain the cause and diagnostic codes of the reset indication.	Issue the X'BF00' output operation to send a reset confirmation packet.
83/4001	Link failure, system starting error recovery for this link.	Issue the X'B100' output operation to end the connection.
83/4002	Connection failure. The diagnostic data parameter will contain more information on this error.	Issue the X'B100' output operation to end the connection.

Return and Reason Codes for X.25 Operation X'BF01'

Table 2-46. Return and Reason Codes for X.25 Operation X'BF01'

Return / Reason Code	Meaning	Recovery
0/0	The X'BF00' output operation was successful. The diagnostic data parameter will contain information indicating if a reset request or reset confirmation packet was sent.	Continue processing.
83/1006	Operation not valid.	Do not issue the X'BF00' output operation on connections that do not support resets.
83/3201	The maximum amount of incoming user data that can be held by user-defined communications support for the user-defined communications application program on this connection has been exceeded.	Wait to receive a failure notification from the QOLRECV program indicating this condition (X'B301' operation, 83/3201 return and reason code). Then, issue the X'B100' output operation to end the connection.
83/3204	Connection ending because a X'B100' output operation was issued.	Wait for notification of the completion of the X'B100' output operation from the QOLRECV program (X'B101' operation).
83/4001	Link failure, system starting error recovery for this link.	Wait to receive a failure notification from the QOLRECV program indicating this condition (X'B301' operation, 83/4001 return and reason code). Then, issue the X'B100' output operation to end the connection.
83/4002	Connection failure.	Wait to receive a failure notification from the QOLRECV program indicating this condition (X'B301' operation, 83/4002 return and reason code). Then, issue the X'B100' output operation to end the connection.

QOLQLIND

```
CALL QOLQLIND(return code,
              reason code,
              number of bytes,
              user buffer,
              line description,
              format)
```

Parameter List

Table 2-47. QOLQLIND Parameter List

Parameter	Use	Type	Description
return code	output	BINARY(4)	Specifies the recovery action to take. See "Return and Reason Codes" on page 2-60.
reason code	output	BINARY(4)	Specifies the error that occurred. See "Return and Reason Codes" on page 2-60.
number of bytes	output	BINARY(4)	Specifies the number of bytes of data returned in the user buffer.
user buffer	output	CHAR(256)	Specifies the buffer where the data from the query will be received. Any unused space in the buffer will be filled with X'00'.
line description	input	CHAR(10)	Specifies the name of the line description to query. An existing token-ring, Ethernet or X.25 line description must be used.
format	input	CHAR(1)	Specifies the format of the data returned in the user buffer. The valid values are as follows: **X'01'** Use format X'01'. See "Format of Data in the User Buffer" for more information.

Description of Function

The QOLQLIND program is called by a user-defined communications application program to query an existing token-ring, Ethernet, or X.25 line description. The data received from the query will be placed in the user buffer parameter.

The line description to be queried need not be associated with any links the user-defined communications application program may have enabled. However, data in the line description may change after it is queried.

Format of Data in the User Buffer

The data received in the user buffer from the query is made up of two parts. The first portion starts at offset 0 from the top of the user buffer and contains general query data. The format of this data does not depend on value of the format parameter supplied to the QOLQLIND program.

Table 2-48. General Query Data

Field	Type	Description
line description	CHAR(10)	Specifies the name of the token-ring, Ethernet or X.25 line description that was queried.
line type	CHAR(1)	Specifies the type of line description that was queried. The valid values are as follows: **X'04'** X.25 **X'05'** Token-ring **X'09'** Ethernet
status	CHAR(1)	Specifies the current status of the line description. The valid values are as follows: **X'00'** Varied off **X'01'** Varied off pending **X'02'** Varied on pending **X'03'** Varied on **X'04'** Active **X'05'** Connect pending **X'06'** Recovery pending **X'07'** Recovery canceled **X'08'** Failed **X'09'** Diagnostic mode **X'FF'** Unknown

The second portion of the user buffer starts immediately after the general query data and contains data specific to the type of line description that was queried. The format of this data depends on the value of the format parameter supplied to the QOLQLIND program.

Token-Ring/Ethernet Specific Data - Format X'01'

Table 2-49 (Page 1 of 2). LAN Specific Data - Format X'01'

Field	Type	Description
local adapter address	CHAR(6)	Specifies, in packed form, the local adapter address of this line. The special value of X'000000000000' indicates that the preset default address for the adapter card was configured. However, the line description must be varied on before this address can be retrieved.
line speed	CHAR(1)	Specifies the speed of this line. The valid values are as follows: **X'01'** 4 megabits/second **X'02'** 10 megabits/second **X'03'** 16 megabits/second
line capability	CHAR(1)	Specifies the capability of this line. The valid values are as follows: **X'00'** Token-ring **X'01'** Ethernet Version 2 **X'02'** Ethernet 802.3 **X'03'** Both Ethernet Version 2 and Ethernet 802.3
line frame size	BINARY(2)	Specifies the maximum frame size possible on this line.
Ethernet Version 2 frame size	BINARY(2)	Specifies the maximum size for Ethernet Version 2 frames. This will be 1502 if the line is capable of Ethernet Version 2 traffic. Otherwise, it will be zero.
number of SSAPs	BINARY(2)	Specifies the number of source service access points (SSAPs) configured for this line.
Note: The following 3 rows are repeated for each SSAP configured for this line.		

Table 2-49 (Page 2 of 2). LAN Specific Data - Format X'01'

Field	Type	Description
SSAP	CHAR(1)	Specifies the configured source service access point.
SSAP type	CHAR(1)	Specifies the SSAP type. The valid values are as follows: **X'00'** Non-SNA SSAP **X'01'** SNA SSAP
SSAP frame size	BINARY(2)	Specifies the maximum frame size allowed on this SSAP.
number of group addresses.	BINARY(2)	Specifies the number of group addresses configured for this line. **Note:** This will always be zero for a token-ring line description.
Note: The following row is repeated for each group address configured for this line.		
group address	CHAR(6)	Specifies a group address, in packed form.

X.25 Specific Data - Format X'01'

Table 2-50 (Page 1 of 2). X.25 Specific Data - Format X'01'

Field	Type	Description
local network address length	CHAR(1)	Specifies, in hexadecimal, the number of binary coded decimal (BCD) digits in the local network address.
local network address	CHAR(9)	Specifies, in BCD, the local network address of this line.
extended network addressing	CHAR(1)	Specifies whether network addressing is extended to permit the use of 17 digits in an address. The valid values are as follows: **X'01'** Network addresses may be up to 15 digits **X'02'** Network addresses may be up to 17 digits
address insertion	CHAR(1)	Specifies if the system inserts the local network address in call request and call accept packets. The valid values are as follows: **'Y'** The local network address is inserted in call request and call accept packets. **'N'** The local network address is not inserted in call request and call accept packets.
modulus	CHAR(1)	Specifies the X.25 modulus value. The valid values are as follows: **X'01'** Modulus 8 **X'02'** Modulus 128
X.25 DCE support	CHAR(1)	Specifies if the system communicates through the X.25 DCE support. This allows the system to communicate with another system without going through an X.25 network. The valid values are as follows: **X'01'** The system does not communicate via the X.25 DCE support **X'02'** The system does communicate via the X.25 DCE support
transmit maximum packet size	BINARY(2)	Specifies the transmit maximum packet size configured for this line.
receive maximum packet size	BINARY(2)	Specifies the receive maximum packet size configured for this line.
transmit default packet size	BINARY(2)	Specifies the transmit default packet size configured for this line.
receive default packet size	BINARY(2)	Specifies the receive default packet size configured for this line.

Table 2-50 (Page 2 of 2). X.25 Specific Data - Format X'01'

Field	Type	Description
transmit default window size	BINARY(1)	Specifies the transmit default window size configured for this line.
receive default window size	BINARY(1)	Specifies the receive default window size configured for this line.
number of logical channels	BINARY(2)	Specifies the number of logical channels configured for this line.
Note: The following 4 rows are repeated for each logical channel configured for this line		
logical channel group number	CHAR(1)	Specifies the logical channel group number. This together with the logical channel number makes up the logical channel identifier.
logical channel number	CHAR(1)	Specifies the logical channel number. This together with the logical channel group number makes up the logical channel identifier.
logical channel type	CHAR(1)	Specifies the logical channel type. The valid values are as follows: **X'01'** Switched virtual circuit (SVC). **X'02'** Permanent virtual circuit (PVC) that is eligible for use by a network controller. **Note:** This does not necessarily mean that this PVC is available for use. Another job running on the network controller attached to this line may already have this PVC in use. **X'22'** PVC that is not eligible for use by a network controller. For example, a PVC that is already attached to an asynchronous controller description.
logical channel direction	CHAR(1)	Specifies the direction of calls allowed on the logical channel. The valid values are as follows: **X'00'** Not applicable (PVC logical channel). **X'01'** Only incoming calls are allowed on this logical channel. **X'02'** Only outgoing calls are allowed on this logical channel. **X'03'** Both incoming and outgoing calls are allowed on this logical channel.

Return and Reason Codes

Table 2-51 (Page 1 of 2). Return and Reason Codes for QOLQLIND

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
81/9999	Internal system error detected. Escape message CPF91F0 will be sent to the user-defined communications application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/1005	Format not valid.	Correct the format parameter. Try the request again.
83/1998	User buffer parameter too small.	Correct the user buffer parameter. Try the request again.
83/2000	Line description not configured for token-ring, Ethernet, or X.25.	Correct the line description parameter. Try the request again.

Table 2-51 (Page 2 of 2). Return and Reason Codes for QOLQLIND

Return / Reason Code	**Meaning**	**Recovery**
83/2002	Not authorized to line description.	Get authorization to the line description. Try the request again.
83/2006	Line description not found.	Correct the line description parameter. Try the request again.
83/2007	Line description damaged.	Delete and recreate the line description. Try the request again.

QOLTIMER

```
          CALL QOLTIMER(return code,
                        reason code,
                        timer set,
                        timer to cancel,
                        queue name,
                        operation,
                        interval,
                        establish count,
                        key length,
                        key value,
                        user data)
```

Parameter List

Table 2-52 (Page 1 of 2). QOLTIMER Parameter List

Parameter	Use	Type	Description
return code	output	BINARY(4)	Specifies the recovery action to take. See "Return and Reason Codes" on page 2-64.
reason code	output	BINARY(4)	Specifies the error that occurred. See "Return and Reason Codes" on page 2-64.
timer set	output	CHAR(8)	Specifies the name of the timer (timer handle) that was set. TIMER001, TIMER002, ... , TIMER128 are the possible values. The content of this parameter is only valid when setting a timer.
timer to cancel	input	CHAR(8)	Specifies the name of the timer (timer handle) to cancel. TIMER001, TIMER002, ... , TIMER128 may be used as values. The special value of *ALL (left-justified and padded on right with spaces) may be used to cancel all timers currently set in the job in which the user-defined communications application program is running. The content of this parameter is only valid when canceling a timer.
queue name	input	CHAR(20)	Specifies the name and library of the data queue where the timer-expired entry will be sent when the timer expires. The first 10 characters specify the name of the data queue and the second 10 characters specify the library in which the data queue is located. Both entries are left-justified. The special values of *LIBL and *CURLIB may be used for the library name. The content of this parameter is only valid when setting a timer.
operation	input	CHAR(1)	Specifies the timer operation to perform. The valid values are as follows: **X'01'** Set a timer. **X'02'** Cancel a timer.

Table 2-52 (Page 2 of 2). QOLTIMER Parameter List

Parameter	Use	Type	Description
interval	input	BINARY(4)	Specifies the number of milliseconds for which to set this timer. Any value between 1,048 and 3,600,000 may be used. The content of this parameter is only valid when setting a timer.
establish count	input	BINARY(4)	Specifies the number of times this timer will be established. Any value between 1 and 60 may be used. The special value of -1 may be used to always have this timer established after it expires. The content of this parameter is only valid when setting a timer.
key length	input	BINARY(4)	Specifies the key length when using a keyed data queue. Any value between 0 and 256 may be used, where 0 indicates the data queue is not keyed. The content of this parameter is only valid when setting a timer.
key value	input	CHAR(256)	Specifies the key value when using a keyed data queue. The content of this parameter is only valid when setting a timer.
user data	input	CHAR(60)	Specifies the user data that is to be included in the timer-expired entry when the timer expires. The content of this parameter is only valid when setting a timer. **Note:** This data is treated as character data only and should not contain pointers.

Description of Function

The QOLTIMER program is called by a user-defined communications application program to either set or cancel a timer. Up to 128 timers, each uniquely identified by a name (timer handle), can be set in the job in which the user-defined communications application program is running.

When the QOLTIMER program is called to set a timer, a timer handle is returned to the user-defined communications application program. The timer handle, along with the user data supplied when the timer was set, is included in the timer-expired entry that is sent to the data queue when the specified amount of time for this timer has elapsed. The timer will then be reestablished, if necessary. For example, suppose a user-defined communications application program sets a timer with a five second interval to be established two times. After five seconds, the timer-expired entry for this timer will be sent to the data queue specified when the timer was set. The timer will then be automatically reestablished, and five seconds later, another timer-expired entry for this timer will be sent to the data queue. See "Timer-Expired Entry" on page 6-5 for the format of the timer-expired entry.

In addition to setting a timer, the user-defined communications application program can call the QOLTIMER program to cancel one or all timers currently set in the job in which the user-defined communications application program is running. User-defined communications support will implicitly cancel a timer in the following cases:

- after a timer has expired the specified number of times
- when a job ends that had one or more timers set

Note: User-defined communications support does not associate timers with links. If necessary, that association must be done by the user-defined communications application program.

Return and Reason Codes

Table 2-53. Return and Reason Codes for QOLTIMER

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
81/9999	Internal system error detected. Escape message CPF91F0 will be sent to the user-defined communications application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/1001	Key length not valid.	Correct the key length parameter. Try the request again.
83/1009	Timer operation not valid.	Correct the operation parameter. Try the request again.
83/1010	Timer interval not valid.	Correct the interval parameter. Try the request again.
83/1011	Number of times to establish timer not valid.	Correct the establish count parameter. Try the request again.
83/3400	Timer not valid on cancel operation.	Correct the timer to cancel parameter. Try the request again.
83/3401	All timers are currently set for the requested set operation.	Cancel a timer. Try the request again.
83/3402	Timer not set on cancel operation.	Continue processing.

Chapter 3. Programming Design Considerations

This chapter discusses concepts related to user-defined communications and how they might relate to the design of a user-defined communications application. This chapter covers:

- Jobs
- Application program feedback
- Programming languages
- X.25 networks
- Token-ring and Ethernet networks
- Data queues
- User spaces

Jobs

A fundamental concept in user-defined communications is the job. A job and a process are the same on the AS/400 system. The terms are used interchangeably in the AS/400 system manuals.

The concept of the job is important because the user-defined communications support performs services for the job requesting the communications support through one of the user-defined communications APIs. Some information used by the user-defined communications support is kept along with other information about the job. This can be displayed with the Work with Job (WRKJOB) command. The Work with Communications Status option displays user-defined communications information for the job (communications handle name, last operation, input and output counts).

A user-defined communications application program always runs within a job. This job may be run interactively or in batch and always represents a separate application to the user-defined communications support. This means that the same protocol can be actively running in more than one job on the system. Also, more than one job can have links that share the same line as other jobs running user-defined communications application programs.

Each link that is enabled by a user-defined communications application program logically consists of the line, network controller, and network device description objects. Many user-defined communications applications can share the same line and controller description, provided the applications are running in different jobs, but each application will use a different device description. Up to 256 device descriptions can be attached to a controller description. This means that there may be a maximum of 256 jobs running user-defined communications applications that share the same line at one time. When an application has finished using a link and disabling it, the network device description used by the application becomes available to another user-defined communications application.

For end-to-end communication to begin, the applications on each system must be started. There is no equivalent function to the intersystem communication function (ICF) program start request. The user-defined communications application program is responsible for providing this support, if needed, To provide this support, the application can have a batch job servicing remote requests to start a user-defined communications application program. This job can be created to run in any subsystem.

For more information on jobs and subsystems, refer to the *Work Management Guide*.

User-defined communications application programs can be designed so that the entire protocol resides within one job or separate jobs where each job represents a portion of the protocol.

There is a one-to-one correspondence between a job and the user-defined communications support for that job. The user-defined communications support for one job does not communicate with the user-defined communications support for another job. If two applications wish to communicate between themselves, a method such as a shared data queue should be used. Also, the data queue could be shared between the two (or more) jobs and the user-defined communications support for those jobs.

Figure 3-1 shows how user-defined communications relate to the AS/400 system job structure and the data queue that provides intraprocess communication between the user-defined communications application and the user-defined communications support.

In this figure, one interactive job is running over an X.25 line (X25USA) to a system in Rochester, Minnesota, using the user-defined communications support. The link was enabled with communications handle name, ROCHESTER.

The user space application programming interfaces (APIs) that the user-defined communications application program is using are shown, along with the data queue programming interface and the user-defined communications support APIs.

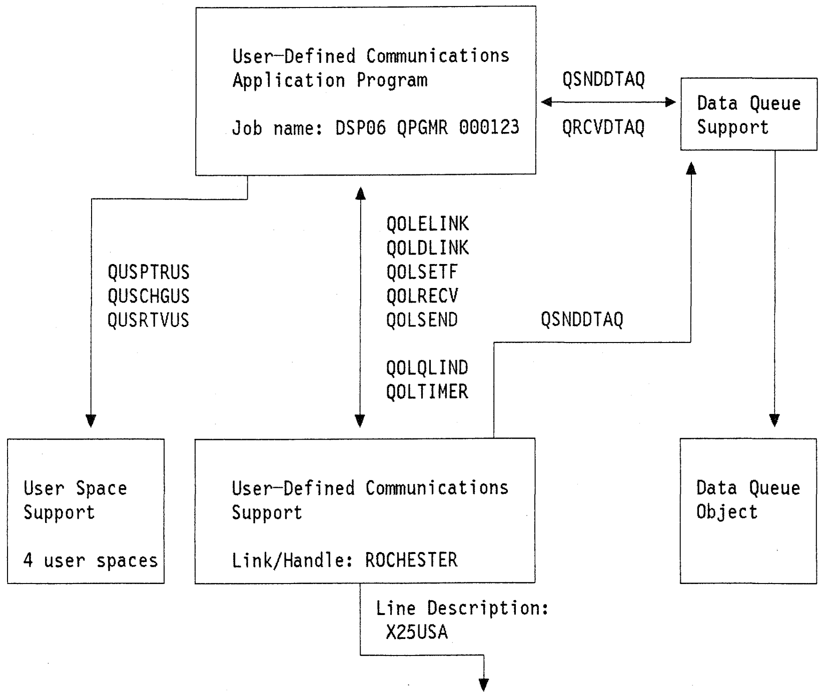
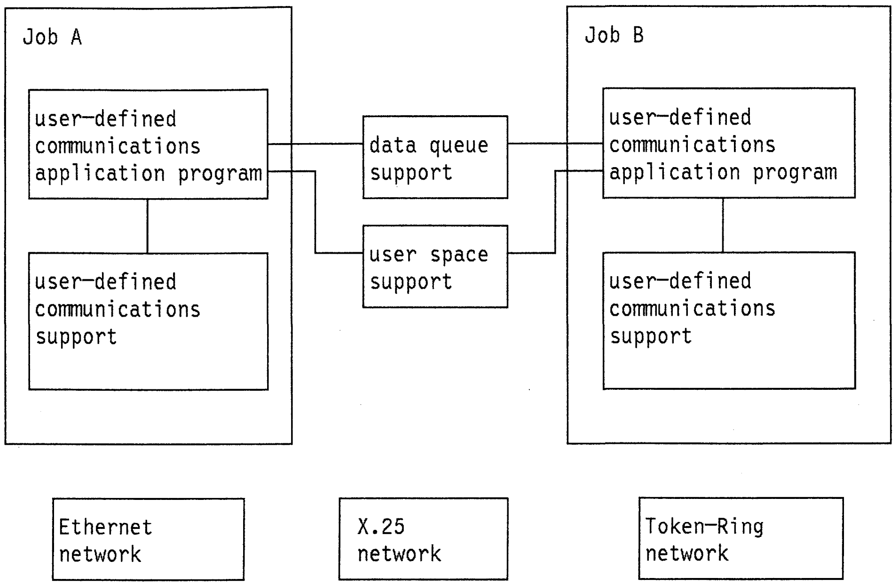


Figure 3-1. Overview of API Relationships

The next figure shows two jobs, Job A and Job B. Each job is using the user-defined communications support to communicate with the networks attached to the AS/400 system by the line description. The figure shows the relationship between the different APIs and the job which is running a user-defined communications application program.

The solid lines in the figure indicate that callable APIs that are used to communicate between the application program and the system services shown.



Job A
user-defined communications application program
user-defined communications support
data queue support
user space support
Job B
user-defined communications application program
user-defined communications support
Ethernet network
X.25 network
Token-Ring network

Figure 3-2. Application Programming Interface to Job Structure

The following list pertains to Figure 3-2.

- A user-defined communications application program uses the data queue APIs, user space APIs, and user-defined communications APIs.
- An application having more than one link enabled can use a separate data queue for each link, or the same data queue for some or all the links that it has enabled.
- The two jobs can communicate with each other using a common data queue. This data queue can be the same data queue that is used for user-defined communications support or a different one.
- The user spaces can be accessed by both jobs, or any other job on the system, with proper authority to the user spaces.
- The user-defined communications support uses the data in the output user spaces that are created when the link was created. The application making the call to QOLSEND can fill the output buffer and descriptor, or another application can do this.
- The user-defined communications support sends data to the application through the input buffer and descriptor that was created when the link on which the data is arriving was created. Either the application making the call to QOLRECV retrieves the data from the input buffer and descriptor, or another application with access to the user spaces does this.

- The application supplies any communications handle (link name) to the link as long as this name is unique among all the other links that the job has enabled.
- An application can enable as many links as there are line descriptions that are supported (X.25, token-ring and Ethernet) and that are able to be varied on.
- An application is able to run over X.25 and LAN links concurrently.

Application Program Feedback

Return and reason codes are used to indicate the success or failure of an operation, and suggested recovery. In severe error conditions an escape message is signaled to the application. If a severe error occurs, user-defined communications will no longer be available to the application.

On the return of QOLSEND and QOLRECV there are cases where the diagnostic field has been filled in. The reason code will indicate if the application should look at the data returned in the diagnostic field.

Synchronous and Asynchronous Operations

Most operations that an application requests on the call to QOLSEND are synchronous operations. Synchronous operations involve one step, which is to call QOLSEND, passing the appropriate information. Synchronous operations complete when QOLSEND returns to the application. The success or failure of the operation is reported in the return and reason codes by QOLSEND.

Asynchronous operations do not complete when QOLSEND returns to the application. There are two steps for every asynchronous operation:

1. A call to QOLSEND to initiate or request the operation.
2. A call to QOLRECV to receive the results of the completed operation.

When QOLSEND returns to the application, the request for the operation has been successfully submitted. After the requested operation has completed, the user-defined communications support sends an incoming data entry (if necessary) to the data queue to instruct the application to call QOLRECV to receive the data. When the application's call to QOLRECV returns, the parameter list contains the success or failure of the operation. If the operation was unsuccessful due to an application template error in the user space used for output, the request is copied into the receive user space. The offset to the template error detected is returned in the parameter list of QOLRECV. Asynchronous operations are only used for X.25 for open connection requests, close connection requests, and resets.

For either type of operation, the application is allowed to reuse the output user spaces as soon as the call to QOLSEND returns.

Programming Languages

User-defined communications support can be called by a program call from any AS/400 system-supported language. Each programming language has advantages and disadvantages of its own, none which directly relate to the user-defined communications support. One consideration for choosing one language over another, is that the programming language needs the ability to set a byte field to any possible hexadecimal value. This does not restrict programming in the different languages, but does make some languages more appealing than others.

Starting and Ending Communications

Relatively little configuration is required to begin communications to the network. For information on configuration, refer to Chapter 6, "Configuration and Additional Information."

To start communications to a network, the user-defined communications application enables the link to the network by calling QOLELINK. Once the link is enabled, a user-defined communications application program can call any of the user-defined communications support APIs, and request any of the supported operations for the link. When the application program has finished communicating with the the network, it disables the link by calling QOLDLINK.

Note: Enabling the link does not result in any communications activity on the network. Disabling a link may cause communications activity for X.25 links if connections are active when the link is disabled.

Programming Considerations for X.25 Applications

The user-defined communications support interface to an X.25 network is at the packet level, which is a connection-oriented level. The user-defined communications application program is responsible for ensuring reliable end-to-end connectivity. End-to-end connectivity means that the user-defined communications application program will initiate, receive, and accept X.25 calls and handle network errors reported to the application, as well as send and receive data.

A user-defined communications application has access to packets which flow over Switched Virtual Circuits (SVCs) and Permanent Virtual Circuits (PVCs). An application can have SVC and PVC connections active concurrently. Up to 32 virtual circuits can be configured on an X.25 line description. The *X.25 Network Guide* provides a good discussion on configuration limitations.

The Display Connection Status (DSPCNNSTS) command displays the virtual circuits that are in use by a the network device, and the state of each connection. This command also displays the active inbound routing information that the application is using to route calls.

X.25 Packet Types Supported

A packet is the basic unit of information transmitted through an X.25 network. In the table below, the X.25 packet types are listed along with the type of service provided. Services for Switched Virtual Circuit (SVC) and Permanent Virtual Circuit (PVC) connections are identified as well as the services that are not accessible (N/A) to a user-defined communications application program.

Packet Type	Application Input or Access	SVC	PVC	N/A
Data	Q,D bits of the general format identifier (GFI) **Note:** The modulus used is configured in the line description. The open connection request allows the user-defined communications support to set the actual window size used.	X	X	

Packet Type	Application Input or Access	SVC	PVC	N/A
Interrupt	32 bytes of data **Note:** On the AS/400 system, the X.25 packet layer provides the confirmation of the receipt of this packet. The call to QOLSEND will not return until the interrupt has been confirmed by the remote system.	X	X	
Reset request	Cause and diagnostic codes **Note:** User-defined communications application provides the confirmation of this packet	X	X	
Reset indication	Cause and diagnostic codes **Note:** User-defined communications application provides the confirmation of this packet	X	X	
Reset confirmation	**Note:** Reset collisions are detected and reported to the application on the reset confirmation	X	X	
Incoming Call	Remote DTE, local virtual circuit, packet and window sizes, up to 109 bytes of additional facilities, up to 128 bytes of bytes of call user data	X		
Call Request	Remote DTE, local virtual circuit, packet and window sizes, up to 109 bytes of additional facilities, up to 128 bytes of bytes of call user data	X		
Call Accept	Packet and window sizes, up to 109 bytes of additional facilities	X		
Call Connected	Negotiated packet and window sizes, facilities	X		
Clear request	Cause and diagnostic codes, facilities, up to 128 bytes of clear user data	X		
Clear indication	Cause and diagnostic codes, facilities, up to 128 bytes of clear user data **Note:** The X.25 packet layer support provides the confirmation on this request	X		
Clear confirmation	**Note:** The X.25 packet layer support provides this support		X	
Receive Ready (RR)	**Note:** The flow of RR and RNR packets is determined by the automatic flow control field of Format I, specified in the open connection request			X
Receive Not Ready (RNR)	**Note:** The flow of RR and RNR packets is determined by the automatic flow control field of Format I, specified in the open connection request			X
Reject (REJ)	**Note:** This packet is not necessarily available on all networks and is not supported by the AS/400 system.			X

Packet Type	Application Input or Access	SVC	PVC	N/A
Restart Request, Indication, and Confirmation	**Note:** These packets affect all virtual circuits on the line			X
Diagnostic	**Note:** This packet is not necessarily available on all networks and is not supported by the AS/400 system.			X
Registration Request and Confirmation	**Note:** This packet is not necessarily available on all networks and is not supported by the AS/400 system.			X

Operations

User-defined communications support defines many different operations. The X'B000' operation will either initiate an X.25 SVC call request, or Open a PVC. By using this operation, an application initiates an open connection request. The X'B100' operation will either initiate an X.25 SVC Clear request (or confirmation), or Close a PVC. By using this operation, an application initiates a close connection request. The open connection request, close connection request, and reset request (or response) operations are two-step operations. Refer to "Synchronous and Asynchronous Operations" on page 3-5 for more information on programming for two-step operations.

The X'B400' operation initiates an X.25 SVC call accept. This operation is known as a call accept operation. The X'0000' operation initiates an X.25 Data packet for a SVC or PVC connection. This operation is called a send data operation. The call accept and send data operations are one-step operations. Refer to "Synchronous and Asynchronous Operations" on page 3-5 for more information on programming for one step operations.

The other X.25 operations are not requested by the application. They are reported to the application in the parameter list of QOLRECV. The X'B201' operation indicates an incoming X.25 SVC call and is known as the call indication operation. The X'B301' operation indicates that a temporary (reset) or permanent (clear) connection failure has occurred. It is known as the connection failure indication operation. Finally, the X'0000' operation indicates incoming data. It is known as the receive data operation.

Connections

User-defined communications support allows X.25 connections over both switched and permanent virtual circuits. A user-defined communications application can have one or many connections active at once. They can be either SVC, PVC, or both. The Display Connection Status (DSPCNNSTS) command shows the state of the connection, logical channel identifier, virtual circuit type, and other information about the call. The states of the connection include activate pending, active, deactivate pending. When the open connection request or call accept operations have not yet completed, the connection state is activate pending. After the open connection request or call accept operations have completed with return and reason codes of zero, the connection state is active. When the close connection request has not yet completed, or if the connection has been cleared by the network, but a close connection request has not been issued by the application, the connection state is deactivate pending.

Note: The connection enters the active state independent of the application receiving the results of the open connection request. Likewise, a connection can become completely close (deactivated, and no longer appears on the DSPCNNSTS screen) independent of the application receiving the results of the close connection request.

In order for the user-defined communications support and the application program to differentiate between connections, connection identifiers are used by both. Connection identifiers are used from the time the connection is started to the time the connection has successfully ended.

Note: A correctly encoded close connection request will always be successful. The only time a close connection request is not successful is when the application coded the close connection request incorrectly. Refer to "Using Connection Identifiers" on page 3-14 for more information.

Connection Identifiers

The user-defined communications support assigns an identifier for each connection. This identifier is reported back to the application as the provider connection end point (PCEP). In the same manner, the user-defined communications application assigns an identifier for each connection and reports it to the communications support as the user connection end point (UCEP). This exchange of identifiers allows both the communications support and the application to refer to a connection in a consistent manner. The UCEP and PCEP are exchanged during the open connection request operation when a PVC is opened or an outgoing call is requested, and the call indication and call accept operations for incoming calls.

The connection is only identified in terms of PCEP and UCEP. For example, the user-defined communications support passes information to the application and reports the UCEP to which the information pertains. In the same manner the user-defined communications application initiates requests for a connection identified by the PCEP.

The user-defined communications support will reuse PCEPs as they become free. PCEPs become free when the application has received notification that the open connection request never completed successfully, or the close connection request has completed successfully. This means that PCEPs are not reused until the applications calls QOLRECV, which returns either of the listed operations. Until the PCEP is freed, data can be received by the application for the UCEP corresponding to the PCEP of the connection.

User-defined communications support places no restrictions on the value of the UCEP, and does not verify its uniqueness. Since all incoming data and connection failure indications are passed to the application using the UCEP connection identifier, the application should ensure uniqueness of each UCEP. Refer to "Using Connection Identifiers" on page 3-14 for information on how to reuse connection identifiers.

Connection Information

In order to ensure reliable end-to-end connectivity, a user-defined communications application keeps track of the control information for each connection it is responsible for. Some of this control information is listed below:

- State of the connection (activating, active, deactivating, reset)
- PCEP for this connection

- SVC or PVC connection indicator
- Negotiated frame sizes; maximum data unit size
- Connection is no longer active indicator or state
- (other application specific information)

The application can use the UCEP as an index into the program's data structures, which keep track of this information.

Switched Virtual Circuit (SVC) Connectivity

Configuration

The SVCs, which are configured in the X.25 line description, are shared among all of the users of the line description. These users are SNA, Asynchronous X.25, OSI, TCP/IP, and user-defined communications. The line description should be defined with enough SVCs to accommodate all of the users of the X.25 line description.

Any SVCs defined in the X.25 line description that are not in use by any controllers (including the network controller) are available to a user-defined communications application program. The available SVCs are distributed as they are requested by the users of the X.25 line description.

Refer to *X.25 Network Guide* for more information on configuring X.25 line descriptions.

For user-defined communications, an SVC is used by an application when the application either initiates a call, or receives an incoming call. The SVC will no longer be in use when the application successfully initiates a clear request to the SVC. Like PVCs, SVCs allow only one application program to have an active connection using the virtual circuit at a time.

Inbound Routing Information

Before an application can receive and accept an incoming call, it must first describe to the user-defined communications support the X.25 calls it wants to look at. The application accomplishes this by issuing a program call to QOLSETF, specifying the inbound routing information in the filter.

The inbound routing information that an application specifies is the first byte of call user data called the protocol ID, or the protocol ID combined with the calling DTE address. In addition, the application specifies whether it will accept calls with fast select, and reverse charging indicated. The calls that the application receives can either be accepted or rejected. The advantage of using filters to reject some calls (based on calling DTE address, fast select, and reverse charging indicated in the incoming call) is that the application is alleviated of some calls which it will always reject.

Once the connection is active, data flows end-to-end between systems and does not need any other technique to route it to the appropriate application.

End-to-End Connectivity

End-to-end connectivity is achieved when one system initiates a call and another accepts the call. When this happens, a connection has been established, and the state of the connection is active. It will remain active until either one of the application programs initiates a clear request, or the network (or system) clears the connection due to an error condition.

Permanent Virtual Circuit (PVC) Connectivity

Configuration

SNA and Asynchronous X.25 controllers use PVCs on the X.25 line by configuring the controller description to logically attach to the PVC. This is not true for users of the network controller description. When a PVC is in use by a user-defined communications application, the system will logically attach the network controller to the PVC. This means that any PVC defined in the X.25 line description and not attached to any controller (including the network controller) is available for use by any user-defined communications application that has a link enabled for the network to which the line is attached.

Because the attaching of PVCs to applications is programmable, one job can have a open a connection over the PVC, end the connection, and then another job can open a different connection over the same PVC. Like SVCs, PVCs allow only one application program to have an active connection using the virtual circuit at a time.

Inbound Routing Information

By definition, the PVC does not require a call to set up a path from one system to another system. As its name suggests, this path always exists (permanent). Therefore, the application has no need to set a filter for the inbound routing information. The inbound data information is automatically routed to the application that has an open connection over the PVC.

End-to-End Connectivity

PVC connections are opened and closed by the application. To open a PVC, the application uses the open connection request operation, just as it does to initiate an X.25 SVC call. To close the PVC, the application uses the close connection request just as it does to clear the SVC call.

Both systems that want to communicate end-to-end must first open the virtual circuit on the local system. When the PVC has been opened on the AS/400 system it is considered active and in use by the application. This is true even if the corresponding remote system doesn't have the virtual circuit active. On the AS/400 system, an open connection request always completes with return and reason codes of zero as long as the PVC is defined in the line description and is not in use by another application. There is no way to detect whether true end-to-end connectivity exists on the PVC.

If the virtual circuit is not active on both systems, and one system attempts to communicate to the other, the virtual circuit on the system with the active PVC will be reset. An application that supports X.25 resets sees the reset arrive as a result of the attempt at sending data. In order to continue, the application responds to the reset. An application that does not support X.25 resets sees a connection failure. The application closes the PVC and opens the PVC in order to continue to use the PVC.

Similarly, when a PVC connection is closed from one system, the other system will see a reset (if reset is supported by that application) or a connection failure if reset is not supported. If the application sees a reset, it must respond to the reset before communications can continue on that connection.

Sending and Receiving Data Packets

Data Sizes

Data units larger and smaller than the negotiated transmit packet size can be sent by a user-defined communications application. Each data unit will be segmented into the appropriate packet sizes by the AS/400 system. Contiguous data larger than the negotiated packet size can also be sent. It will be divided into individual packets and sent out with the more-data-indicator on. The user-defined communications application program should request that the data unit size be a multiple of the transmit and receive packet sizes configured in the line description. The user-defined communications application program sets important values that pertain to each connection. See "X.25 SVC and PVC Output Operations" on page 2-22 for important information on these values, which are contained in the user spaces, and the importance of these values to the user-defined communications application.

The values that the application supplies should be carefully determined and tailored to the needs of the application. Similarly, the application uses the values returned to ensure negotiated limitations are not exceeded.

The application uses three values in determining how to fill the output user-space buffer. These values are:

- Maximum data unit size
- Maximum data unit assembly size
- Negotiated transmit packet size

The maximum data unit size is the value that the application specifies when the link is enabled. The maximum data unit assembly size is the total length of contiguous input data that is supported by the application. Contiguous data units have the more data indicator set on in all the data units in the sequence; except the last data unit, which has the more data indicator set off. The application indicates the maximum data unit assembly size on the open connection request. The maximum data unit assembly size should always be greater than the maximum data unit size, to make full use of the user spaces. The negotiated transmit packet size is returned when the open connection request completes. The application uses these values together to determine how to fill in the user space output buffer.

Note: If the remote system exceeds the maximum data unit assembly size and sends the sequence to the local AS/400 system, the connection is closed or cleared.

Refer to "Maximum Amount of Outstanding Data" on page 3-13 for related information on incoming data limitations.

Interrupts: The Interrupt is a special data packet. A restriction is imposed by an X.25 network specifying that a DTE cannot have more than one outstanding Interrupt on any virtual call in each direction. A user-defined communications application issues an Interrupt by calling QOLSEND. The QOLSEND program will not return to the application program until the Interrupt confirmation has been received. It is important to understand the Interrupt procedures of the remote DTE in order to avoid any wait conditions due to the remote system not sending the confirmation to the Interrupt.

Flow Control

The Receive Ready (RR) and Receive Not Ready (RNR) packets are sent by the AS/400 system on behalf of the user-defined communications application. The distribution of these packets is based on the automatic flow control field in the open connection request operation. The automatic flow control (RR/RNR) is sent to prevent one system from overrunning another system with data.

When the automatic flow control value is exceeded for a connection because a remote system is sending data at a rate too fast for the local system, an RNR packet is sent on behalf of the application on that local system. Once the application on the local system receives the data, an RR will be sent to allow more data to be received by the local system's communications support.

The automatic flow control value should be set high enough so that RR/RNR processing occurs so frequently that performance on the virtual circuit is affected, and low enough that the application can process the data fast enough. If an application is coded properly, the RR and RNR processing is not noticed by the application, just as for other system users of X.25.

To avoid situations where the virtual circuit is not operational because an RNR was sent, or to avoid excessive amounts of RR and RNR processing, the application program should always attempt to receive all the data from the communications support. An incorrectly coded application can cause another application to wait indefinitely for an RR to open the virtual circuit for communications. If the applications are coded correctly, the RR and RNR packet sequences are unnoticed by the applications.

Maximum Amount of Outstanding Data

The communications support will set aside a limited amount of data for the application(s) it is servicing. For X.25 it is 128K for each connection. If the 128K limitation is met, an error log entry is generated and the connection is cleared (SVCs) or reset (PVCs) by the system. Before this occurs, the AS/400 system attempts to slow the incoming data traffic by issuing an RNR on behalf of the application. An RR is sent after the application has received one-third of the amount of outstanding data.

Refer to "Flow Control" for related information on incoming data and flow control.

Reset Support

When a user-defined communications application initiates a reset, it is also responsible for discarding any data that the user-defined communications support has received. The user-defined communications support will only discard data if the connection is closed.

Using Connection Identifiers

The following figures illustrate how to use connection identifiers. The figures illustrate how the two step operations, open connection request, and close connection request relate to the UCEP and PCEP identifiers. Special attention is paid to the outstanding two-step operations. This is important so that the application can correctly interpret the PCEP and reuse UCEPs.

The connections in each figure refer to SVC connections. The same principles apply when using PVC connections.

Outgoing Connection Scenarios

The following figures show how the application program handles UCEPs and PCEPs for outgoing connections.

Example 1

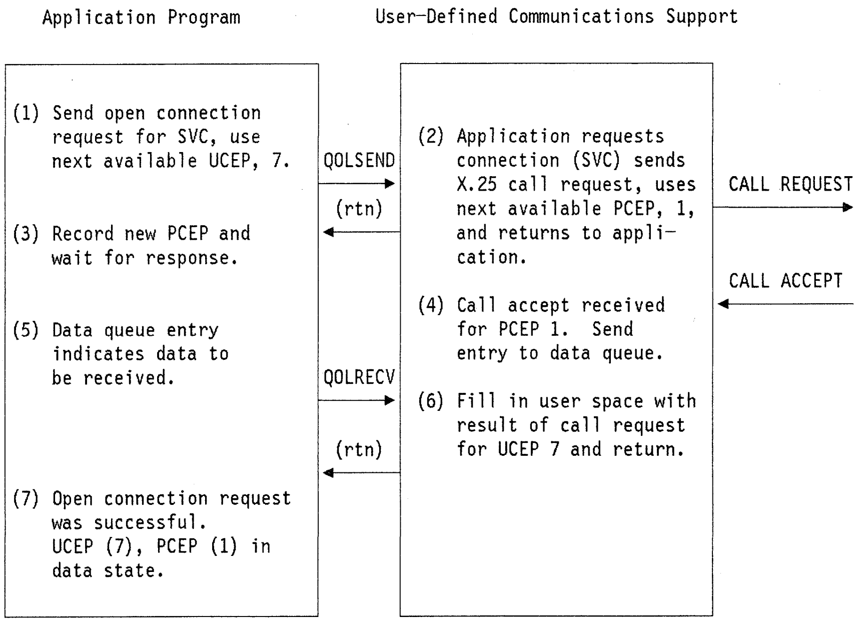

Application Program
User-Defined Communications Support
(1) Send open connection request for SVC, use next available UCEP, 7.
QOLSEND
(rtn)
(2) Application requests connection (SVC) sends X.25 call request, uses next available PCEP, 1, and returns to application.
CALL REQUEST
(3) Record new PCEP and wait for response.
CALL ACCEPT
(4) Call accept received for PCEP 1. Send entry to data queue.
(5) Data queue entry indicates data to be received.
QOLRECV
(6) Fill in user space with result of call request for UCEP 7 and return.
(rtn)
(7) Open connection request was successful. UCEP (7), PCEP (1) in data state.

Figure 3-3. Normal Connection Establishment

1. The application wishes to open a connection, so it calls QOLSEND passing it the UCEP it wants to use for the connection. The application keeps track of the UCEP, PCEP pair. At this point UCEP=7, and PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP (1); and returns to the application, acknowledging the receipt of the request.

   The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1. The UCEP=7, PCEP=1 connection is not yet active. Next, the application calls QRCVDTAQ to wait for the incoming data entry. The application is expecting the open connection response.
4. The X.25 call accept is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.

5. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.
6. The user-defined communications support fills in the input user space with data for the open connection response operation, and determines the UCEP that the data is for by using the PCEP that the X.25 call accept is for. Since the call accept was received for PCEP=1, the UCEP is 7.
7. The application's call to QOLRECV returns with successful return and reason codes for the open connection response operation. This operation was reported for UCEP 7; the UCEP=7, and PCEP=1 connection is now active.

Example 2

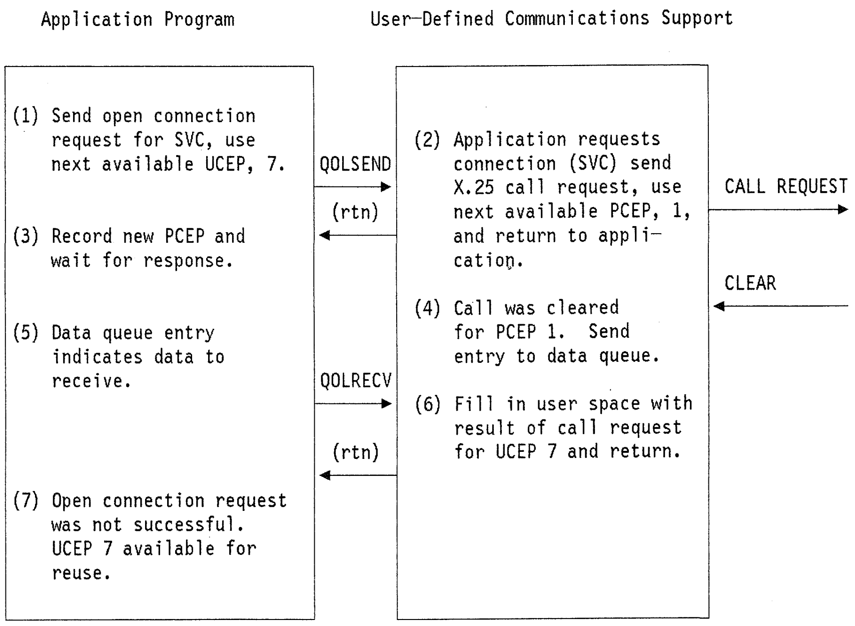

Application Program
User-Defined Communications Support
(1) Send open connection request for SVC, use next available UCEP, 7.
QOLSEND
(2) Application requests connection (SVC) send X.25 call request, use next available PCEP, 1, and return to application.
CALL REQUEST
(rtn)
(3) Record new PCEP and wait for response.
CLEAR
(4) Call was cleared for PCEP 1. Send entry to data queue.
(5) Data queue entry indicates data to receive.
QOLRECV
(6) Fill in user space with result of call request for UCEP 7 and return.
(rtn)
(7) Open connection request was not successful. UCEP 7 available for reuse.

Figure 3-4. Connection Request Cleared by Network/Remote System

1. The application wishes to open a connection, so it calls QOLSEND, passing it the UCEP it wants to use for the new connection. The application keeps track of the UCEP, PCEP pair. At this point UCEP=7, and PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP (1); and returns to the application, acknowledging the receipt of the request.

   The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls QRCVDTAQ to wait for the incoming data entry. The application is expecting the open-connection response.
4. A clear is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
5. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.
6. The user-defined communications support fills in the input user space with data for the open connection response operation, and determines the UCEP

for the data by using the PCEP for which the X.25 call accept was received. Since the call was cleared for PCEP=1, the UCEP is 7. The PCEP=1 is no longer active, and may be reused by the user-defined communications support.

7. The application's call to QOLRECV returns with unsuccessful return and reason codes for the open connection response operation. This for PCEP=1, the UCEP is 7. The PCEP=1 is no longer active, and operation is for UCEP=7. Because the connection is not open, the user-defined communications support's PCEP=1 no longer implies UCEP=7, and the application's UCEP=7 may be reused.

Example 3

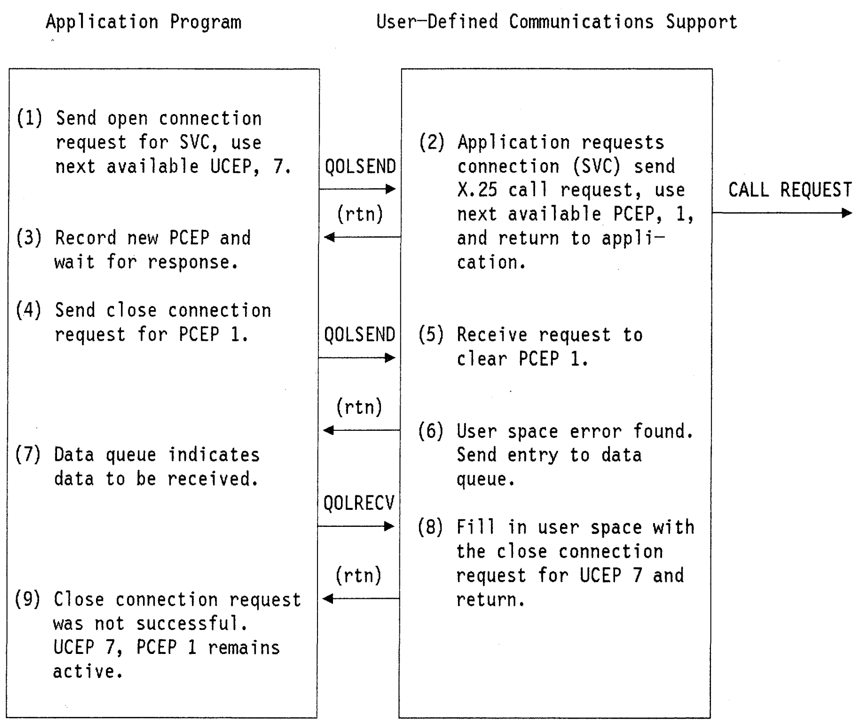

Application Program
User-Defined Communications Support
(1) Send open connection request for SVC, use next available UCEP, 7.
QOLSEND
(rtn)
(2) Application requests connection (SVC) send X.25 call request, use next available PCEP, 1, and return to application.
CALL REQUEST
(3) Record new PCEP and wait for response.
(4) Send close connection request for PCEP 1.
QOLSEND
(5) Receive request to clear PCEP 1.
(rtn)
(6) User space error found. Send entry to data queue.
(7) Data queue indicates data to be received.
QOLRECV
(8) Fill in user space with the close connection request for UCEP 7 and return.
(rtn)
(9) Close connection request was not successful. UCEP 7, PCEP 1 remains active.

Figure 3-5. Request to Clear Connection with Outstanding Call (Unsuccessful)

1. The application wishes to open a connection, so it calls QOLSEND passing it the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point UCEP=7, and PCEP is undefined.

2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP (1); and returns to the application, acknowledging the receipt of the request.

   The user-defined communications support validates the request and issues the X.25 call request.

3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls QRCVDTAQ to wait for the incoming data entry. The application is expecting the open connection response.

4. The application no longer wants the UCEP=7 connection. It calls QOLSEND passing the PCEP=1, to identify the connection to be closed.

5. The user-defined communications support receives the request, and returns to the application, acknowledging the receipt of the request.

   The user-defined communications support validates the request and finds an error.

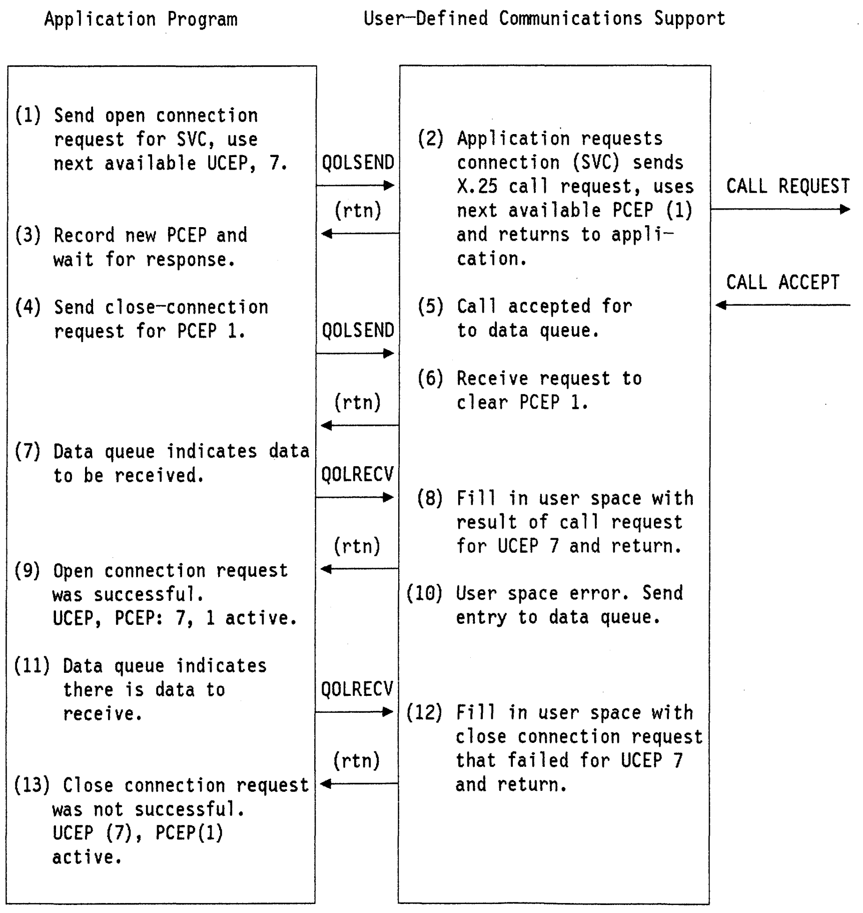
6. The user space error is found. A copy of the user space in error will be passed back to the application. To inform the application of the unsuccessful close connection request, an incoming data entry is sent to the data queue.

7. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.

8. The user-defined communications support fills in the input user space with data for the unsuccessful close connection request operation, and determines the UCEP that the data is for by using the PCEP that the close connection was requested for. Since the close connection request was for PCEP=1, the UCEP is 7. The PCEP=1 is no longer active.

9. The application's call to QOLRECV returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is still in use by both the application and the user-defined communications support. The application can either correct the error and reissue the operation, or wait for the call to be accepted or rejected.

Example 4



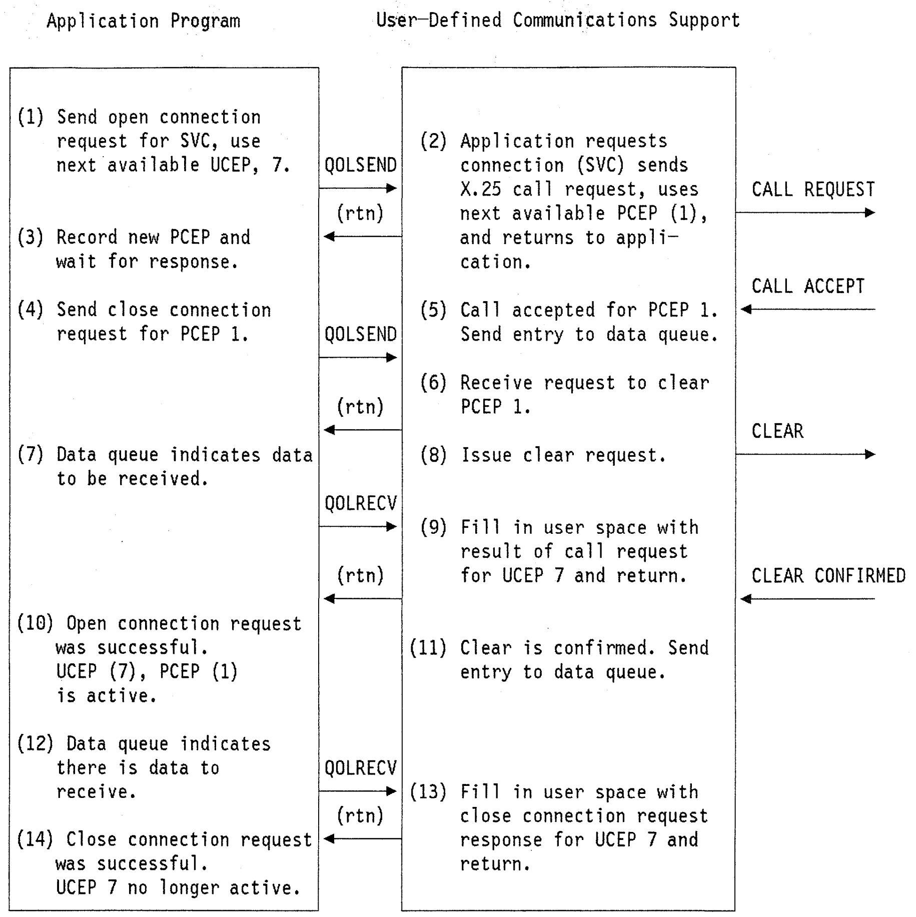
Application Program
User–Defined Communications Support
(1) Send open connection request for SVC, use next available UCEP, 7.
QOLSEND
(2) Application requests connection (SVC) sends X.25 call request, uses next available PCEP (1) and returns to application.
CALL REQUEST
(rtn)
(3) Record new PCEP and wait for response.
CALL ACCEPT
(4) Send close–connection request for PCEP 1.
(5) Call accepted for to data queue.
QOLSEND
(6) Receive request to clear PCEP 1.
(rtn)
(7) Data queue indicates data to be received.
QOLRECV
(8) Fill in user space with result of call request for UCEP 7 and return.
(rtn)
(9) Open connection request was successful. UCEP, PCEP: 7, 1 active.
(10) User space error. Send entry to data queue.
(11) Data queue indicates there is data to receive.
QOLRECV
(12) Fill in user space with close connection request that failed for UCEP 7 and return.
(rtn)
(13) Close connection request was not successful. UCEP (7), PCEP(1) active.

Figure 3-6. Unsuccessful Attempt to Clear Outstanding (Successful) Call

1. The application wishes to open a connection, so it calls QOLSEND, passing the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point UCEP=7, and PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP (1); and returns to the application, acknowledging the receipt of the request.

   The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls QRCVDTAQ to wait for the incoming data entry. The application is expecting the open connection response.
4. The application no longer wants the UCEP=7 connection. It calls QOLSEND passing the PCEP=1, to identify the connection to be closed.
5. The X.25 call accept is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
6. The user-defined communications support receives the request, and returns to the application, acknowledging the receipt of the request.
7. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.
8. The user-defined communications support fills in the input user space with data for the open connection response, and determines the UCEP that the data is for by using the PCEP for the X.25 call accept. Since the call accept was received for PCEP=1, the UCEP is 7.
9. The application's call to QOLRECV returns with successful return and reason codes for the open connection request operation. This operation is reported for UCEP=7; the UCEP=7, PCEP=1 connection is now active with an outstanding close connection request.
10. While processing the close connection request, the user-defined communications support detects an error in the user space. The user space that is in error is copied into the input user space, so the application is aware of the data in error. To inform the application of the unsuccessful close connection request, an incoming data entry is sent to the data queue.
11. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.
12. The user-defined communications support fills the input user space with data for the unsuccessful close connection request operation, and determines the UCEP that the data is for by using the PCEP that was requested for the close connection was requested. Since the close connection request was for PCEP=1, the UCEP is 7. The PCEP=1 is still active.
13. The application's call to QOLRECV returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is still in use by both the application and the user-defined communications support. The application can either correct the error and reissue the operation, or wait for the call to be accepted or rejected.

Example 5



(1) Send open connection request for SVC, use next available UCEP, 7.

(2) Application requests connection (SVC) sends X.25 call request, uses next available PCEP (1), and returns to application.

(3) Record new PCEP and wait for response.

(4) Send close connection request for PCEP 1.

(5) Call accepted for PCEP 1. Send entry to data queue.

(6) Receive request to clear PCEP 1.

(7) Data queue indicates data to be received.

(8) Issue clear request.

(9) Fill in user space with result of call request for UCEP 7 and return.

(10) Open connection request was successful. UCEP (7), PCEP (1) is active.

(11) Clear is confirmed. Send entry to data queue.

(12) Data queue indicates there is data to receive.

(13) Fill in user space with close connection request response for UCEP 7 and return.

(14) Close connection request was successful. UCEP 7 no longer active.

Figure 3-7. Successful Attempt to Clear Outstanding (Successful) Call

1. The application wishes to open a connection so it calls QOLSEND, passing it the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point UCEP=7, and PCEP is undefined.

2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP (1); and returns to the application, acknowledging the receipt of the request.

   The user-defined communications support validates the request and issues the X.25 call request.

3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls QRCVDTAQ to wait for the incoming data entry. The application is expecting the open connection response.

4. The application no longer wants the UCEP=7 connection. It calls QOLSEND passing the PCEP=1, to identify the connection to be closed.

5. The X.25 call-accept is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.

6. The user-defined communications support receives the request, and returns to the application, acknowledging the receipt of the request.

7. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.

8. The user-defined communications support validates the close connection request, and issues an X.25 Clear request.

9. The user-defined communications support fills in the input user space with data for the open connection response, and determines the UCEP that the data is for by using the PCEP for the X.25 call accept. Since the call accept was received for PCEP=1, the UCEP is 7.

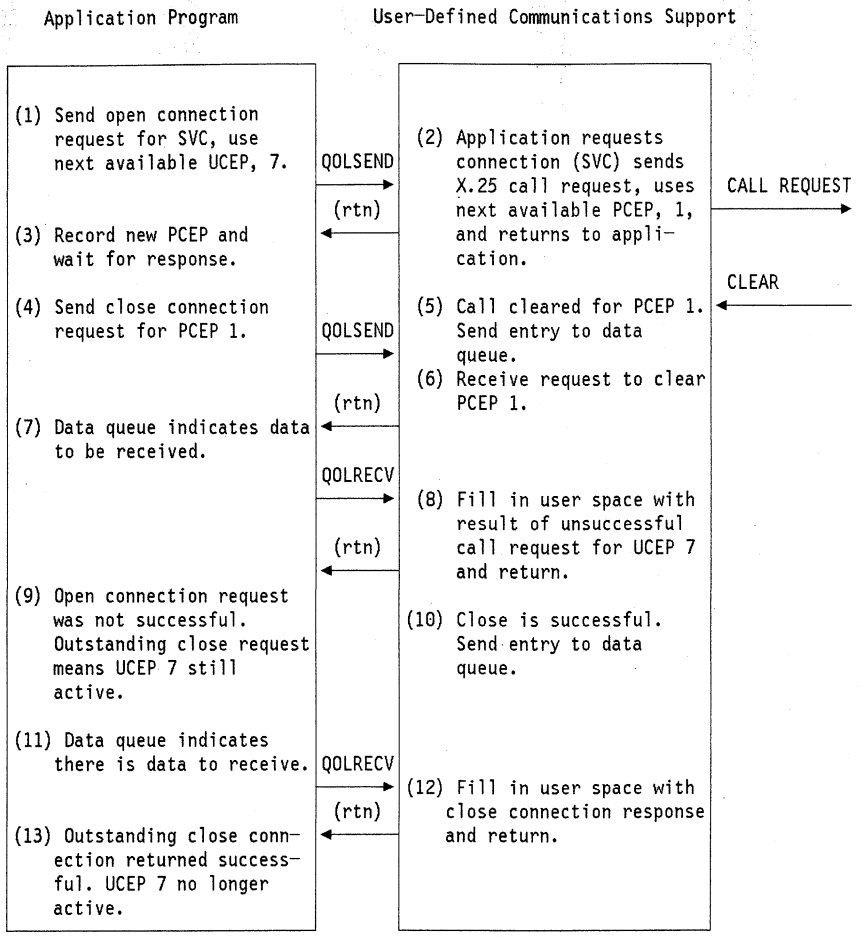
10. The application's call to QOLRECV returns with successful return and reason codes for the open connection request operation. This operation is reported for UCEP=7; the UCEP=7, PCEP=1 connection is now active with an outstanding close connection request.

11. The clear confirmation is received for PCEP=1. To inform the application of the successful close connection request, an incoming data entry is sent to the data queue.

12. The user-defined communications support fills the input user space with data for the successful close connection request operation, and determines the UCEP that the data is for by using the PCEP that was requested for the close connection. Since the close connection request was for PCEP=1, the UCEP is 7. The PCEP=1 is no longer active.

13. The application's call to QOLRECV returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is no longer active.

Example 6



(1) Send open connection request for SVC, use next available UCEP, 7.

(2) Application requests connection (SVC) sends X.25 call request, uses next available PCEP, 1, and returns to application.

(3) Record new PCEP and wait for response.

(4) Send close connection request for PCEP 1.

(5) Call cleared for PCEP 1. Send entry to data queue.

(6) Receive request to clear PCEP 1.

(7) Data queue indicates data to be received.

(8) Fill in user space with result of unsuccessful call request for UCEP 7 and return.

(9) Open connection request was not successful. Outstanding close request means UCEP 7 still active.

(10) Close is successful. Send entry to data queue.

(11) Data queue indicates there is data to receive.

(12) Fill in user space with close connection response and return.

(13) Outstanding close connection returned successful. UCEP 7 no longer active.

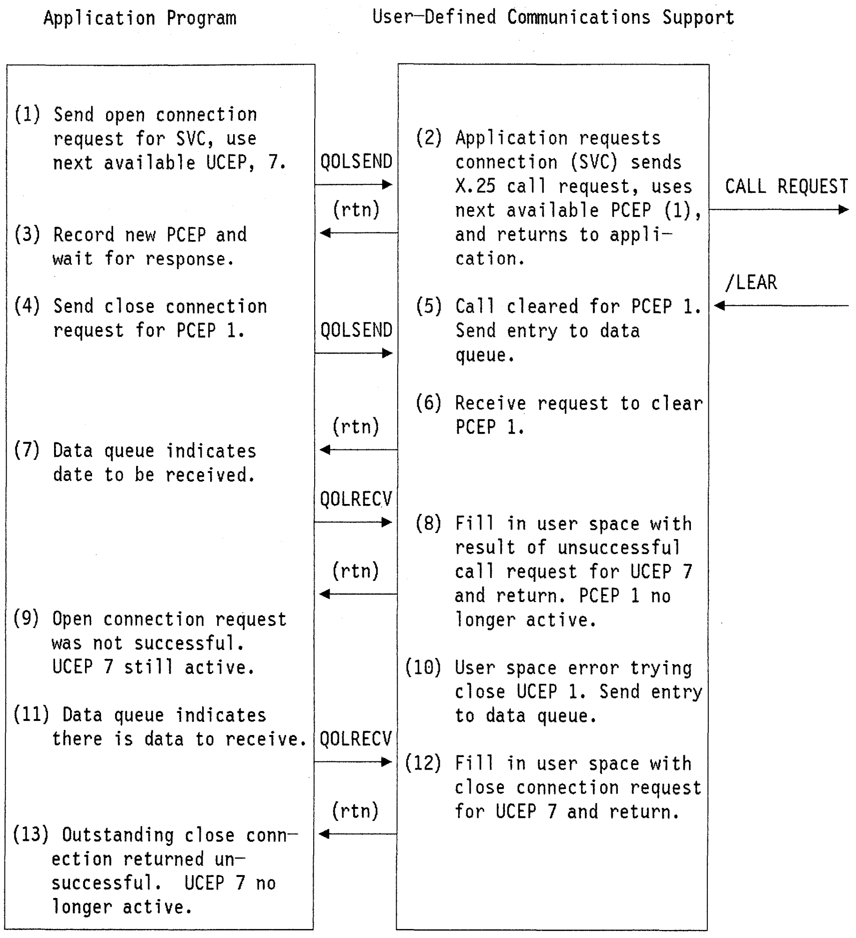
Figure 3-8. Successful Attempt to Clear Outstanding (Unsuccessful) Call

1. The application wishes to open a connection, so it calls QOLSEND, passing it the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point UCEP=7, and PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP (1); and returns to the application, acknowledging the receipt of the request.

   The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls QRCVDTAQ to wait for the incoming data entry. The application is expecting the open connection response.
4. The application no longer wants the UCEP=7 connection. It calls QOLSEND passing the PCEP=1, to identify the connection to be closed.
5. The X.25 Clear is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
6. The user-defined communications support receives the request, and returns to the application, acknowledging the receipt of the request.

7. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.
8. The user-defined communications support fills in the input user space with data for the open connection response, and determines the UCEP that the data is for by using the PCEP that the X.25 call accept is for. Since the call accept was received for PCEP=1, the UCEP is 7.
9. The application's call to QOLRECV returns with unsuccessful return and reason codes for the open connection request operation. This operation is reported for UCEP=7. Because the close connection request is outstanding, the UCEP=7, PCEP=1 connection is not fully closed.
10. The request is validated, but no clear is sent because the connection was cleared previously. The close is considered successful, and an entry is sent to the data queue.
11. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.
12. The user-defined communications support fills in the input user space with data for the successful close connection request operation, and determines the UCEP that the data is for by using the PCEP that the close connection was requested for. Since the close connection request was for PCEP=1, and the UCEP is 7. The PCEP=1 is no longer active.
13. The application's call to QOLRECV returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is no longer active.

Example 7


Application Program

User–Defined Communications Support

(1) Send open connection request for SVC, use next available UCEP, 7.

QOLSEND

(2) Application requests connection (SVC) sends X.25 call request, uses next available PCEP (1), and returns to application.

CALL REQUEST

(rtn)

(3) Record new PCEP and wait for response.

/LEAR

(4) Send close connection request for PCEP 1.

QOLSEND

(5) Call cleared for PCEP 1. Send entry to data queue.

(6) Receive request to clear PCEP 1.

(rtn)

(7) Data queue indicates date to be received.

QOLRECV

(8) Fill in user space with result of unsuccessful call request for UCEP 7 and return. PCEP 1 no longer active.

(rtn)

(9) Open connection request was not successful. UCEP 7 still active.

(10) User space error trying close UCEP 1. Send entry to data queue.

(11) Data queue indicates there is data to receive.

QOLRECV

(12) Fill in user space with close connection request for UCEP 7 and return.

(rtn)

(13) Outstanding close connection returned unsuccessful. UCEP 7 no longer active.

Figure 3-9. Unsuccessful Attempt to Clear Outstanding (Unsuccessful) Call

1. The application wishes to open a connection, so it calls QOLSEND passing it the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point UCEP=7, and PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP (1); and returns to the application, acknowledging the receipt of the request.

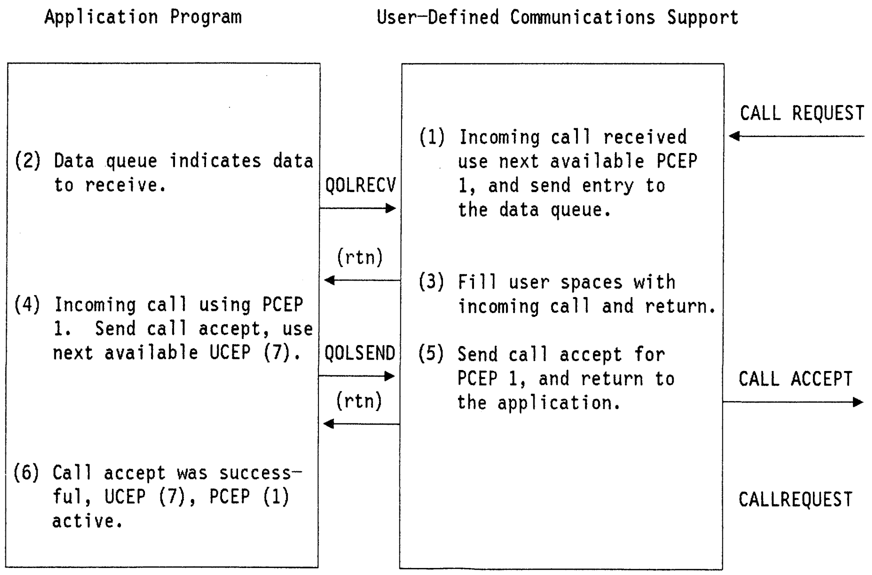
   The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls QRCVDTAQ to wait for the incoming data entry. The application is expecting the open connection response.
4. The application no longer wants the UCEP=7 connection. It calls QOLSEND passing the PCEP=1, to identify the connection to be closed.
5. The X.25 Clear is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
6. The user-defined communications support receives the request, and returns to the application, acknowledging the receipt of the request.

7. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.
8. The user-defined communications support fills in the input user space with data for the open connection response, and determines the UCEP that the data is for by using the PCEP that the X.25 call accept is for. Since the call accept was received for PCEP=1, the UCEP is 7.
9. The application's call to QOLRECV returns with unsuccessful return and reason codes for the open connection request operation. This operation is reported for UCEP=7. Because the close connection request is outstanding, the UCEP=7, PCEP=1 connection is not fully closed.
10. The request is validated, and an error is found in the user space. An entry is sent to the data queue.
11. The application's call to QRCVDTAQ returns with the incoming data entry. The application then issues a call to QOLRECV.
12. The user-defined communications support fills in the input user space with data for the unsuccessful close connection request operation, and determines the UCEP that the data is for by using the PCEP that the close connection was requested for. Since the close connection request was for PCEP=1, the UCEP is 7. Because the connection was cleared prior to the close connection request, the PCEP=1, UCEP=7 connection is considered no longer active to the user-defined communications support.
13. The application's call to QOLRECV returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is no longer active.

Incoming Connections

The following figures show how the application program handles UCEPs and PCEPs for incoming connections.

Example 1



Application Program
User-Defined Communications Support
CALL REQUEST
(1) Incoming call received use next available PCEP 1, and send entry to the data queue.
(2) Data queue indicates data to receive.
QOLRECV
(rtn)
(3) Fill user spaces with incoming call and return.
(4) Incoming call using PCEP 1. Send call accept, use next available UCEP (7).
QOLSEND
(5) Send call accept for PCEP 1, and return to the application.
CALL ACCEPT
(rtn)
(6) Call accept was successful, UCEP (7), PCEP (1) active.
CALLREQUEST

Figure 3-10. Normal Connection Establishment

1. An incoming call request is received by the communications support, which determines there is an application that has a filter satisfying this call request. The communications support uses the next available PCEP=1 for this new connection. An entry is sent to the data queue.
2. The application has been waiting for its call to QRCVDTAQ to complete. The call completes indicating there is data to be received. The application calls QOLRECV.
3. The input user space is filled with the incoming call request for PCEP=1, and QOLRECV returns.
4. The application looks at the operation, which indicates an incoming call indication. The PCEP reported by the communications support is 1. The application chooses to accept this call, and passes the UCEP to be used for this new connection. The call is made to QOLSEND with PCEP=1, UCEP=7.
5. The call accept is received and sent for PCEP=1. QOLSEND returns to the application.
6. The call accept request was successful for UCEP=7, PCEP=1. This connection is now active.

Example 2

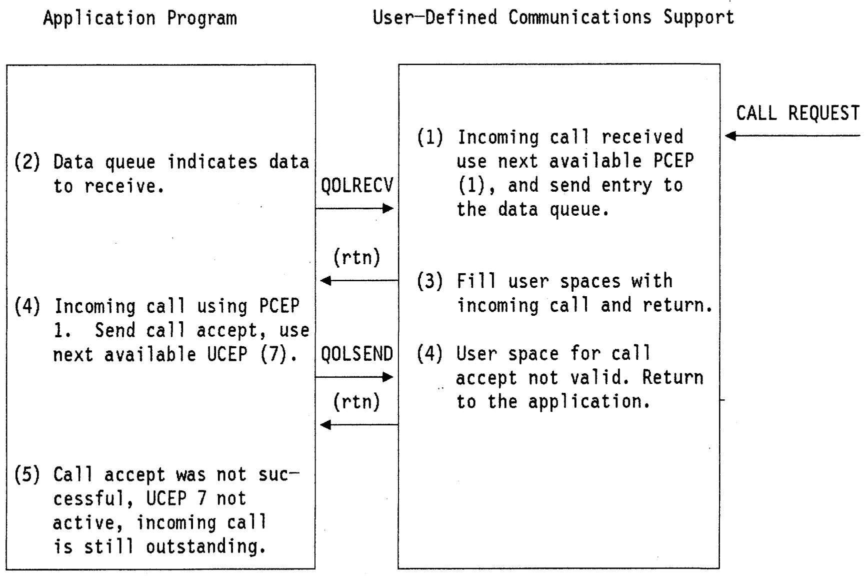

Application Program
User-Defined Communications Support
CALL REQUEST
(1) Incoming call received use next available PCEP (1), and send entry to the data queue.
(2) Data queue indicates data to receive.
QOLRECV
(rtn)
(3) Fill user spaces with incoming call and return.
(4) Incoming call using PCEP 1. Send call accept, use next available UCEP (7).
QOLSEND
(4) User space for call accept not valid. Return to the application.
(rtn)
(5) Call accept was not successful, UCEP 7 not active, incoming call is still outstanding.

Figure 3-11. Send call accept not Valid

1. An incoming call request is received by the communications support, which determines there is an application that has a filter satisfying this call request. The communications support uses the next available PCEP=1 for this new connection. An entry is sent to the data queue.
2. The application has been waiting for its call to QRCVDTAQ to complete. It does, indicating there is data to be received. The application calls QOLRECV.
3. The input user space is filled with the incoming call request for PCEP=1, and QOLRECV returns.
4. The application looks at the operation which indicates an incoming call indication. The PCEP reported by the communications support is 1. The appli-

cation chooses to accept this call, and passes the UCEP to be used for this new connection. The call is made to QOLSEND with PCEP=1, UCEP=7.

5. The call accept is received and an error is found in the user space. QOLSEND returns to the application, reporting the error and offset. The incoming call is still outstanding for PCEP=1.
6. The application checks the return and reason codes and finds that an error has occurred. The call accept was not sent and the incoming call is still waiting for a response.

Example 3

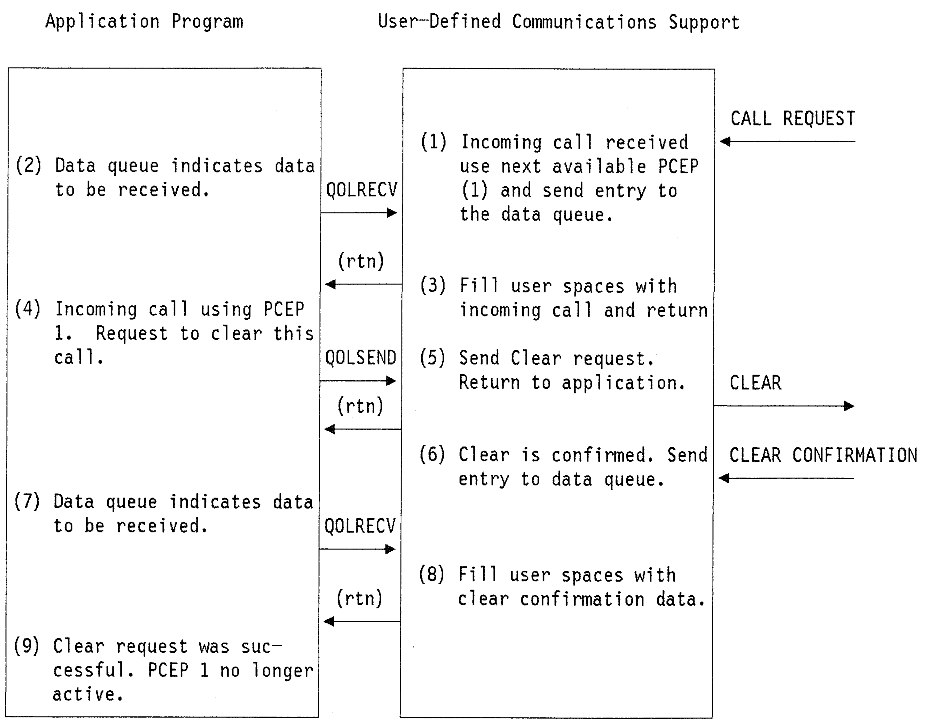

Application Program
User-Defined Communications Support
CALL REQUEST
(1) Incoming call received use next available PCEP (1) and send entry to the data queue.
(2) Data queue indicates data to be received.
QOLRECV
(rtn)
(3) Fill user spaces with incoming call and return
(4) Incoming call using PCEP 1. Request to clear this call.
QOLSEND
(5) Send Clear request. Return to application.
CLEAR
(rtn)
(6) Clear is confirmed. Send entry to data queue.
CLEAR CONFIRMATION
(7) Data queue indicates data to be received.
QOLRECV
(8) Fill user spaces with clear confirmation data.
(rtn)
(9) Clear request was successful. PCEP 1 no longer active.

Figure 3-12. Send Clear for Incoming Call

1. An incoming call request is received by the communications support, which determines there is an application that has a filter satisfying this call request. The communications support uses the next available PCEP=1 for this new connection. An entry is sent to the data queue.
2. The application has been waiting for its call to QRCVDTAQ to complete. It does, indicating there is data to be received. The application calls QOLRECV.
3. The input user space is filled for the incoming call request for PCEP=1, and QOLRECV returns.
4. The application looks at the operation which indicates an incoming call indication. The PCEP reported by the communications support is 1. The application does not wish to accept the call, so the user space is filled in for a close connection request and the application calls QOLSEND
5. The close connection request is received and QOLSEND returns to the application, acknowledging the request.

   The close connection request is validated and a clear is sent.

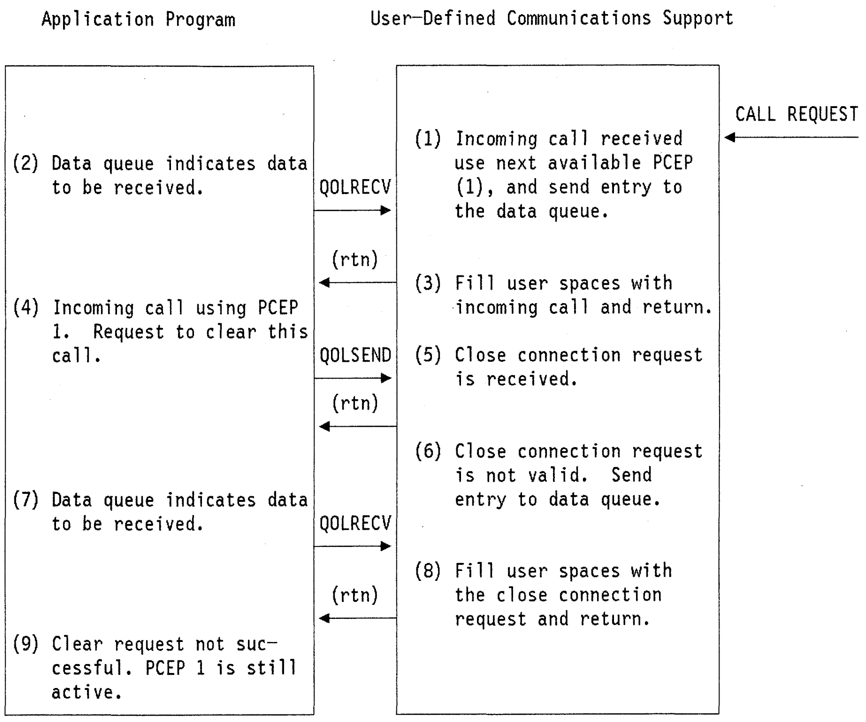
6. The clear confirmation is received for PCEP=1 which has no UCEP. An incoming data entry is sent to the data queue.

7. The application's call to QRCVDTAQ returns with the incoming data entry. The application calls QOLRECV to receive the data.

8. The input user space is filled in with the clear confirmation data. Since the connection was never established (and the application never assigned a UCEP to this connection), QOLRECV returns to the application passing a UCEP of 0.

9. The close connection request was successful. PCEP=1 is no longer active.

Example 4



Application Program
User-Defined Communications Support
CALL REQUEST
(1) Incoming call received use next available PCEP (1), and send entry to the data queue.
(2) Data queue indicates data to be received.
QOLRECV
(rtn)
(3) Fill user spaces with incoming call and return.
(4) Incoming call using PCEP 1. Request to clear this call.
QOLSEND
(5) Close connection request is received.
(rtn)
(6) Close connection request is not valid. Send entry to data queue.
(7) Data queue indicates data to be received.
QOLRECV
(8) Fill user spaces with the close connection request and return.
(rtn)
(9) Clear request not successful. PCEP 1 is still active.

Figure 3-13. Send Clear for Incoming Call

1. An incoming call request is received by the communications support, which determines there is an application that has a filter satisfying this call request. The communications support uses the next available PCEP=1 for this new connection. An entry is sent to the data queue.

2. The application has been waiting for its call to QRCVDTAQ to complete. It completes indicating there is data to be received. The application calls QOLRECV.

3. The input user space is filled for the incoming call request for PCEP=1, and QOLRECV returns.

4. The application looks at the operation which indicates an incoming call indication. The PCEP reported by the communications support is 1. The application does not wish to accept the call, so the user space is filled in for a close connection request and QOLSEND

5. The close connection request is received and QOLSEND returns to the application, acknowledging the request.

6. The close connection request is validated and an error is found. An entry is sent to the data queue.
7. The application's call to QRCVDTAQ return, with the incoming data entry. The application calls QOLRECV to receive the data.
8. The input user space is filled in with the unsuccessful close request, and QOLRECV returns to the application.
9. The close connection request was not successful. UCEP=7, PCEP=1 is still active.

Closing Connections

The following figures show how the application program closes a connection. The figures apply to both incoming and outgoing connections.

The next two figures illustrate that a close connection request never completely guarantees the connection will be closed.

Example 1

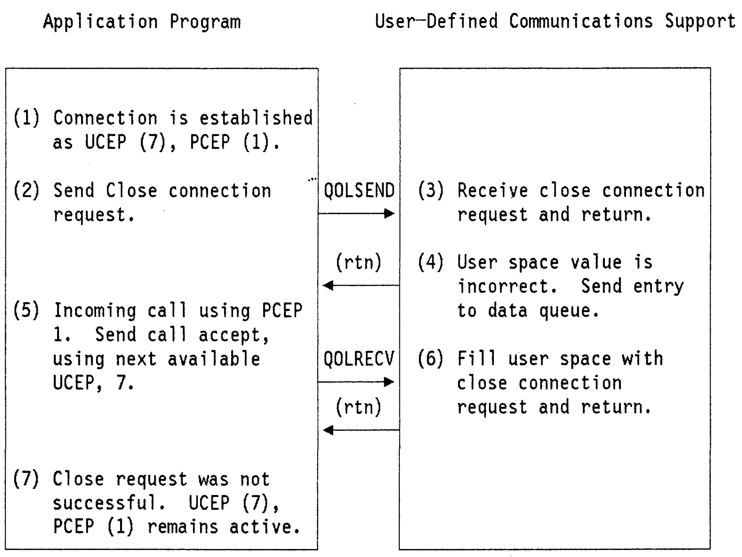

Application Program
User-Defined Communications Support
(1) Connection is established as UCEP (7), PCEP (1).
(2) Send Close connection request.
QOLSEND
(3) Receive close connection request and return.
(rtn)
(4) User space value is incorrect. Send entry to data queue.
(5) Incoming call using PCEP 1. Send call accept, using next available UCEP, 7.
QOLRECV
(6) Fill user space with close connection request and return.
(rtn)
(7) Close request was not successful. UCEP (7), PCEP (1) remains active.

Figure 3-14. Close Connection Request is Not Valid

Example 2

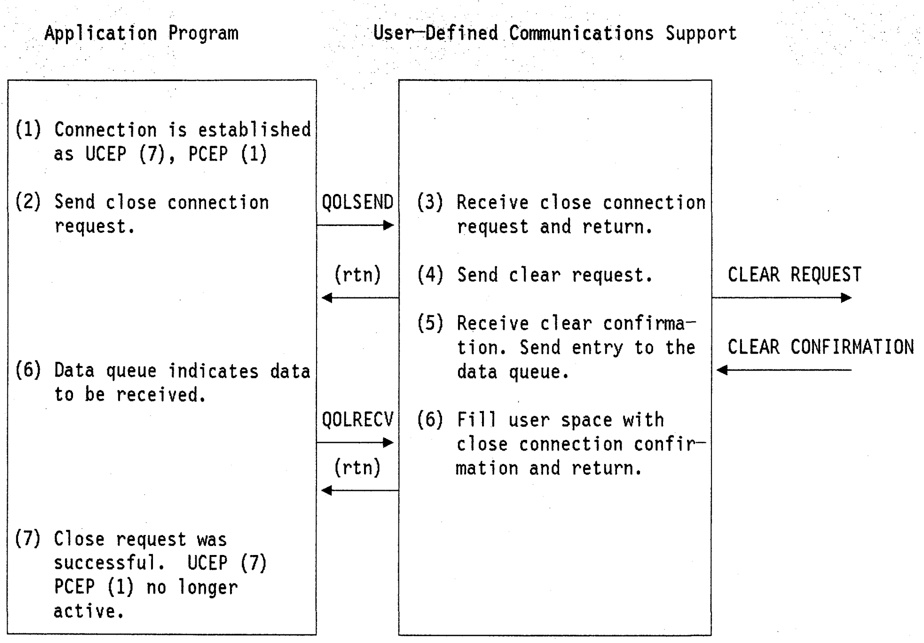

Application Program
User-Defined Communications Support
(1) Connection is established as UCEP (7), PCEP (1)
(2) Send close connection request.
QOLSEND
(3) Receive close connection request and return.
(rtn)
(4) Send clear request.
CLEAR REQUEST
(5) Receive clear confirmation. Send entry to the data queue.
CLEAR CONFIRMATION
(6) Data queue indicates data to be received.
QOLRECV
(6) Fill user space with close connection confirmation and return.
(rtn)
(7) Close request was successful. UCEP (7) PCEP (1) no longer active.

Figure 3-15. Close Connection Request is Valid

AS/400 System X.25 Call Control

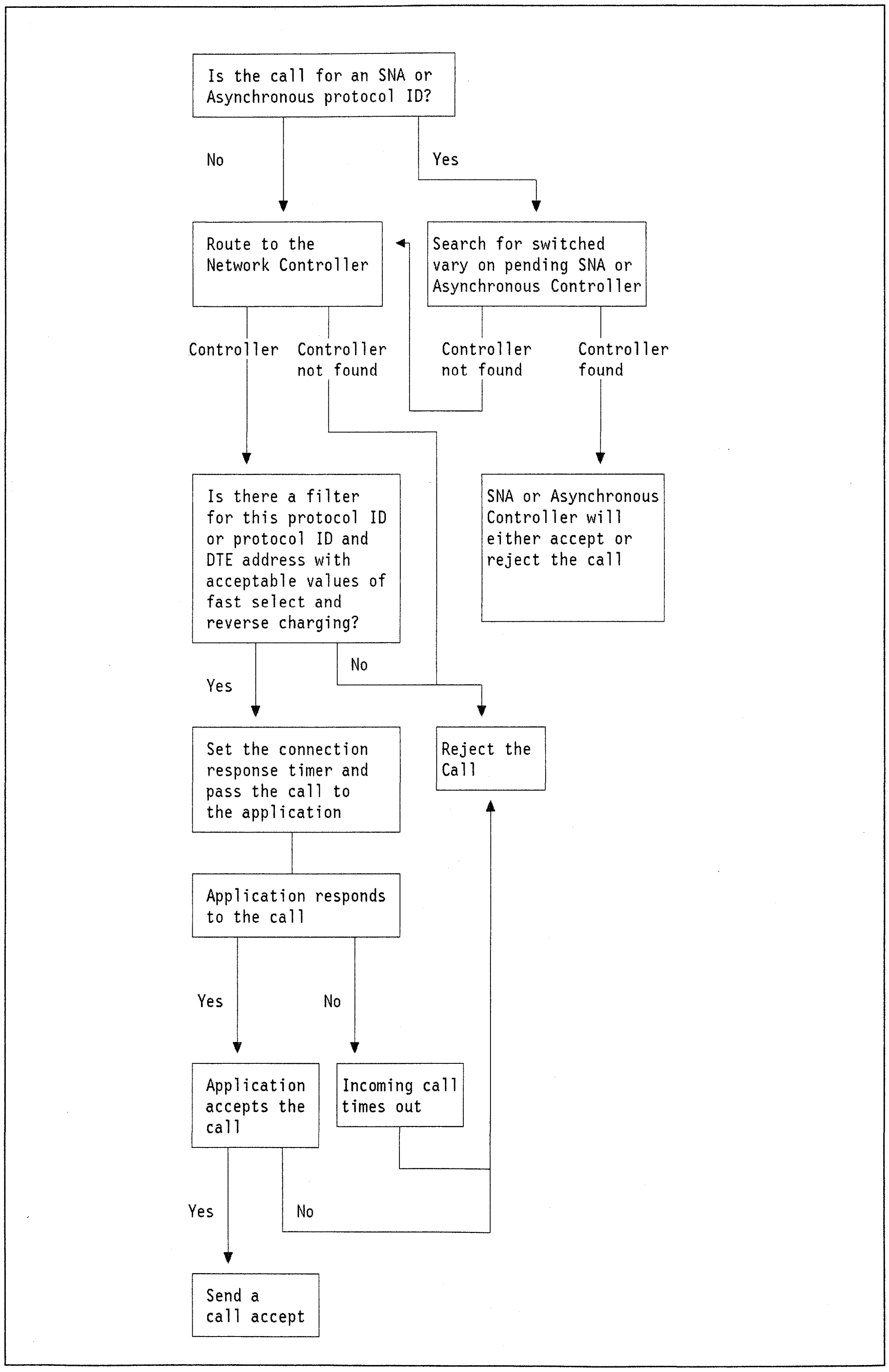
X.25 calls arriving to the AS/400 system are routed by the X.25 support, based primarily on the protocol ID field. This field is the first byte of call-user data in the X.25 call packet.

Protocol IDs are formally defined and used by SNA, Asynchronous X.25, and TCP/IP protocols. The fact that some protocol IDs are defined by other communications protocols does not prevent a user-defined communications application from using these same protocol IDs to either initiate or accept X.25 calls. Refer to the *X.25 Network Guide* for a list of protocol identifiers supported by communications subsystems on the AS/400 system.

AS/400 system X.25 call control is described as SNA/Asynchronous-X.25-first. This means that if an incoming call contains an SNA or Asynchronous X.25 protocol ID, the system will attempt to route the call to SNA or Asynchronous X.25 support. The system will attempt to route the call to the appropriate controller for SNA or Asynchronous X.25. If there is no switched vary on pending SNA or Asynchronous X.25 controller to accept the call, it will be routed to the network controller.

The network controller will search for a filter to match the protocol ID (and calling DTE address). If no filter exists, the call is rejected by the system. If a filter exists for the protocol ID (or protocol ID and calling DTE address) the call and filter is further checked for matching fast select and reverse charging data. If the call specifies reverse charging, but reverse charging is not specified in the filter, the call is rejected with an X'42' diagnostic code. If the call specifies fast select, but fast select is not included in the filter, the call is rejected with an X'A4' diagnostic code.

Note: TCP/IP, OSI, and user-defined communications support all share the same network controller and specify inbound routing information to route the calls to their applications. If two different applications (for example, TCP/IP and a user-defined communications application) request the same inbound routing information, the first to request the filter will be granted the filter. The second request will be rejected, indicating that a filter is already in use for the inbound routing information specified in the request.



Is the call for an SNA or Asynchronous protocol ID?
No
Yes
Route to the Network Controller
Search for switched vary on pending SNA or Asynchronous Controller
Controller
Controller not found
Controller not found
Controller found
Is there a filter for this protocol ID or protocol ID and DTE address with acceptable values of fast select and reverse charging?
SNA or Asynchronous Controller will either accept or reject the call
No
Yes
Set the connection response timer and pass the call to the application
Reject the Call
Application responds to the call
Yes
No
Application accepts the call
Incoming call times out
Yes
No
Send a call accept

Figure 3-16. X.25 Call Control Algorithm

Performance Considerations

It is important to realize that the X'0000' operation is completely synchronous. This means that control is not returned to the application until all the data passed in the data units are sent and confirmed by the DCE. Some implications of this are:

- If the application sends data on a connection that has data flow turned off (the remote system sent an RNR to the local AS/400 system), a subsequent call to QOLSEND with operation X'0000' will not return until the remote system sends the RR to turn flow back on for the connection.
- When transmitting Interrupt packets, control is not returned to the application until the interrupt is confirmed by the remote DTE. If the remote DTE is an AS/400 system, the interrupt is confirmed by the AS/400 system X.25 packet layer support. If the network is congested, the use of Interrupt packets may cause a decrease in performance for the application.

In these situations, it may be appropriate to have one job for each connection (each virtual circuit).

Programming Considerations for LAN Applications

LAN Frames Supported

User-defined communications over LANs use connectionless (unacknowledged) service. Unacknowledged Information (UI) frames are the only frames a user-defined communications application can generate.

The following tables show the user-defined communications application program's interface to the LAN frames.

Table 3-1. Ethernet Frame Format

UI Frame Field	Description	No Access
Preamble	Synchronization field	X
Destination Address (DA)	Specifies the MAC adapter address of the remote station	
Source Address (SA)	Specifies the MAC adapter address of the local station	X
Type	Specifies the upper layer protocol used. For example, TCP/IP uses X'0800' and X'0806', SNA uses X'80D5'.	
Information	User data	
Frame Check Sequence (FCS)	Used for cyclic redundancy checking	X

Table 3-2. Ethernet 802.3 Frame Format

UI Frame Field	Contents & Application Access	No Access
Preamble	Synchronization field	X
Start Frame Delimiter (SFD)	Bit pattern that indicates the beginning of the frame	X
Destination Address (DA)	Specifies the MAC adapter address of the remote station	
Source Address (SA)	Specifies the MAC adapter address of the local station	X
Length	Length of the data portion of the frame	
Destination Service Access Point (DSAP)	Indicates the upper layer protocol the frame is for	
Source Service Access Point (SSAP)	Indicates the upper layer protocol the frame is from	
Control	Contains special indicators	X
Information	User data	
Pad	Used for to pad data frames less than 48 bytes	X
Frame Check Sequence (FCS)	Used for cyclic redundancy checking	X

Table 3-3. Token-Ring 802.5 Frame Format

UI Frame Field	Contents & Application Access	No Access
Starting Delimiter (SD)	Bit pattern that indicates the beginning of the frame	X
Access Control	Contains priority, token, monitor and reserved bits. **Note:** The application can only access the priority bits of this field.	
Frame Control	Determines MAC or LLC frame	X
Destination Address (DA)	Specifies the MAC adapter address of the remote station	
Source Address (SA)	Specifies the MAC adapter address of the local station	X
Routing Information	Routing information supplied for frames intended for a system that is not directly attached to the token-ring network.	
Destination Service Access Point (DSAP)	Indicates the upper layer protocol the frame is for	
Source Service Access Point (SSAP)	Indicates the upper layer protocol the frame is from	
Control	Contains control information	X
Information	User data	
Frame Check Sequence (FCS)	Used for cyclic redundancy checking	X
Ending Delimiter	Contains error/control information	X
Frame Status Field	Contains control information	X

Operations

User-defined communications support defines many different operations. Not all operations are valid on all data links. The operations which are valid for LAN links are:

- X'0000' and X'0001'. These operations together represent the send- and receive-data operations for any of the LAN frames types.

Configuration

The Service Access Point (SAP) that the user-defined application uses to send and receive data must be configured in the line description. The 04, 06, and AA SAPs are automatically generated in the line description unless the SAPs are manually configured. The 04 SAP is used by SNA, and the 06 and AA SAPs are used by TCP/IP. An application can choose to use any SAP (including SAPs defined by SNA or IEEE). The line description must be configured to include the SAPs the application uses. The SAPTYPE for each SAP used must be configured as *NONSNA.

Although it is possible to use any SAP configurable on the AS/400 system, it is not recommended to use SNA SAPs for user-defined communications, since this may restrict the use of SNA on your AS/400 system. In the same manner, using the same SAP as other well-known protocols, such as TCP/IP, may restrict the use of these protocols or the user-defined communications application on the AS/400 system.

Note: It is not possible to run an SNA application and a user-defined communications application over the same SAP concurrently. It is possible to run a TCP/IP application and a user-defined communications application over the same SAP concurrently, provided the inbound routing information is unique among all the non-SNA applications sharing the network controller.

Inbound Routing Information

For the user-defined communications application to receive data from a LAN, it must inform the communications support of how to filter the inbound data and route it to the application. This is accomplished by a program call to QOLSETF. The fields in the incoming frame which are used to route the data are DSAP, SSAP, MAC address, and type.

The different filters allow the user-defined application to distinguish its data from the rest of the data on the LAN. The more selective the inbound routing information is, the less chance there is that the application will be processing unnecessary input requests. Also, more selective inbound routing information allows multiple jobs running user-defined communications applications to share the same SAP.

For example, if an application is using 92 SSAP and 92 DSAP but only talking to one remote system, it may want to set a more selective filter which would include DSAP, SSAP, and MAC address. Conversely, if an application accepts data on the 04 SAP from systems sending data on any SAP, then the application would set a filter for DSAP only, indicating that it will accept all data arriving on the 04 SAP.

End-to-End Connectivity

Because user-defined communications on a LAN is connectionless, it is up to the user-defined communications application protocol to define a method to reach the remote systems it communicates with. There are several ways to do this. One way is to have each system configured in a database file on the AS/400 system. Each system could have a local name that the application program uses to correlate with the MAC address and routing information. LANs provide a technique to broadcast, which can be used to retrieve this information as well. An example of this is the Address Resolution Protocol (ARP) used by TCP/IP, which returns the MAC address and routing information so that a system without that information can communicate with a new remote system.

Sending and Receiving Data

Maximum Frame Size

The data unit size created by the user-defined communications support is always large enough to contain the maximum frame size supported by any of the SAPs configured for non-SNA use (SAPTYPE(*NONSNA)). This value is returned in the parameter list on the call to QOLELINK, and is called the maximum data unit size. For the Ethernet (802.3) and token-ring, the maximum frame size that is configured in the line description will be the maximum frame size that can be specified by the application. There is no minimum frame size for the Ethernet 802.3 or token-ring LANs.

Ethernet Version 2 does not define SAPs for the higher-layer protocols. Therefore, the maximum frame size is not determined by the maximum frame size for a SAP. The maximum frame size for Ethernet Version 2 is 1502. The first 2 bytes are for the type field, and the last 1500 bytes are for user data. The minimum amount of data that it can send is 48 bytes. The first 2 bytes for the type field, and the next 46 bytes are for user data. If the line is configured to handle both Ethernet 802.3 and Ethernet Version 2 data, the larger of the configured value and 1502 is chosen and reported to the application on the maximum data unit size parameter returned from QOLELINK.

If the user-defined communications application attempts to send data frames which are larger or smaller than supported, the output request completes with nonzero return and reason codes and an error code is returned to the application in the diagnostic information field.

User-defined communications application accesses information that is contained in the line description through the QOLQLIND API. It is best to make the call to QOLQLIND after the link has been successfully enabled because the application is assured that the information passed in the QOLQLIND parameter list is accurate for as long as the link is enabled. The application uses the information on the frame size for the SAP to send the correct amount of data over the SAP.

Maximum Amount of Outstanding Data

Most often, the data will arrive at a slightly faster rate than the application can receive it. The communications support will keep data intended for an application so that the application can receive it. However, there is a limit to the amount of data that can be kept for the application later. This is to avoid one system from overrunning another system's resources. If this limit is reached, all new incoming data frames for that application will be discarded until the application picks up one third of the data which has been stored for the application. Since the data consists of unacknowledged information frames, the higher-layer protocol within the user-defined application detects the loss of data, resends the data, or performs other recovery actions.

Each time the limit is exceeded, the communications support generates an error log entry and puts a message in the QSYSOPR message queue, indicating that the unacknowledged service has temporarily failed.

Ethernet to Token-Ring Conversion and Routing

The IBM 8209 Ethernet to token-ring bridge provides additional connectivity options for the AS/400 system. Refer to *IBM 8209 LAN Bridge Customer Information*, SA21-9994, for more details.

Performance Considerations

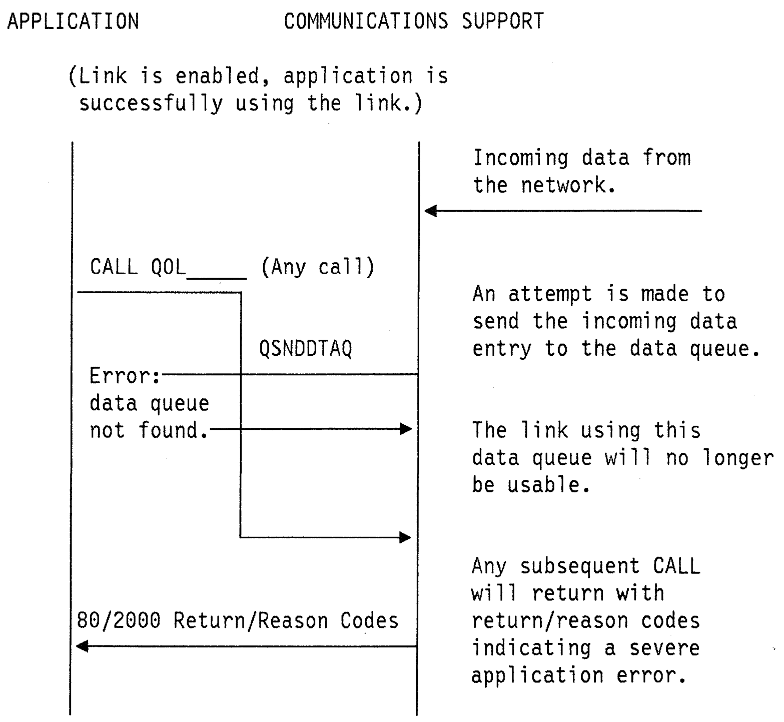
The user-defined communications application program enables connectionless traffic to enter the AS/400 system from the LAN. In the call to QOLSETF, the DSAP parameter indicates the SAP which will be activated on the AS/400 system. By activating traffic over a SAP, data will be taken from the LAN and brought into the AS/400 system. Similarly, deactivating traffic over a SAP causes traffic intended for that SAP to be left on the LAN and not brought into the AS/400 system.

To keep the token-ring and Ethernet line speeds optimal, the SAP or SAPs that the user-defined communications application uses should be deactivated as soon as the application no longer wants to receive traffic for the SAP. If the link is disabled and no other applications are using the SAP(s), they are deactivated automatically by the user-defined communications support.

Protocols that use broadcast frames as a discovery technique could flood the network with messages and affect performance on all the systems attached to the network.

Data Queue Considerations

A user-defined communications application will use a data queue for intraprocess communications between the application and the communications support to take place. The data queue should be provided by the application prior to the call to QOLELINK. The link will never be fully enabled if the data queue does not exist. Communications will no longer be available if the user-defined communications support detects that the data queue has been deleted.


APPLICATION
COMMUNICATIONS SUPPORT
(Link is enabled, application is successfully using the link.)
Incoming data from the network.
CALL QOL____ (Any call)
An attempt is made to send the incoming data entry to the data queue.
QSNDDTAQ
Error: data queue not found.
The link using this data queue will no longer be usable.
Any subsequent CALL will return with return/reason codes indicating a severe application error.
80/2000 Return/Reason Codes

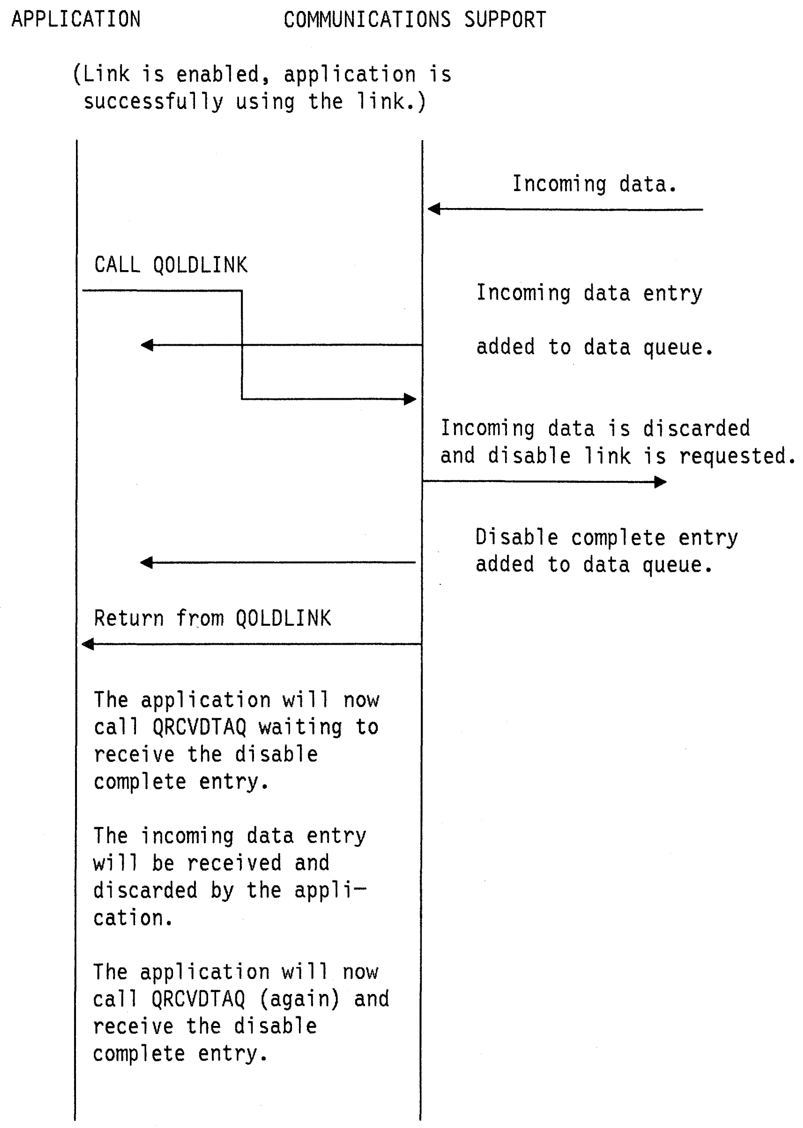
In addition to using the data queue for intraprocess communications between the application and the communications support, the application can use the data queue to provide interprocess communications with other applications.

The data queue can be keyed so that each process receives data queue entries only for it's key. This allows both jobs to put two kinds of entries on the data queue. One for intraprocess communication and the other for interprocess communication. The key can also serve as a way to prioritize entries on the data queue.

The content of any of the data queue entries that are defined and used by the application is not restricted by the user-defined communications support. User-defined communications support will never attempt to receive any entries from the data queue. Therefore, it is up to the application to receive the entries from the data queue and perform the appropriate actions for that entry based on the handle (or timer handle) the entry is for.

This means that it may be necessary for the application to clear the old messages from the data queue, if the data queue is to be reused. For example, if a link is disabled, all communication entries for that link (denoted by the communication handle) prior to the disable complete entry are no longer valid. Because timer support does not depend on the communications support, timer entries will still be valid.

The following scenario shows that an incoming data entry received by the application is no longer valid because the application made a request to disable the link.



APPLICATION COMMUNICATIONS SUPPORT

(Link is enabled, application is successfully using the link.)

Incoming data.

CALL QOLDLINK

Incoming data entry added to data queue.

Incoming data is discarded and disable link is requested.

Disable complete entry added to data queue.

Return from QOLDLINK

The application will now call QRCVDTAQ waiting to receive the disable complete entry.

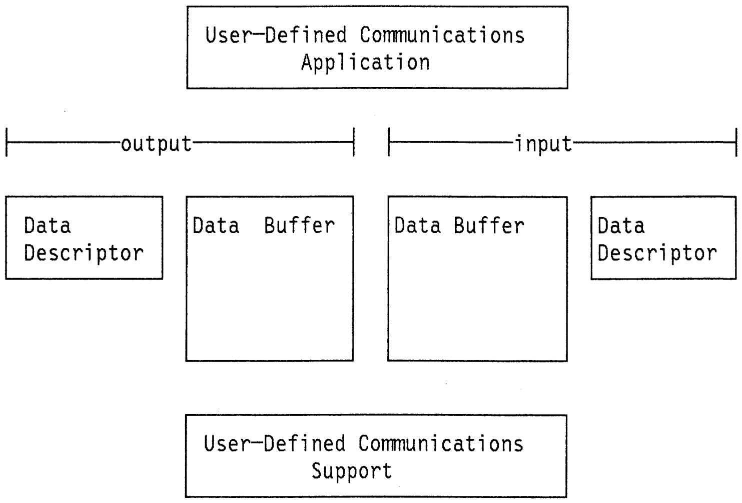
The incoming data entry will be received and discarded by the application.

The application will now call QRCVDTAQ (again) and receive the disable complete entry.

User Space Considerations

The application will use user space objects (*USRSPC) to hold the input and output buffers and descriptors. The AS/400 system provides application programming interfaces that can be used to manipulate the user spaces.

The user spaces are created by the user-defined communications support as part of an enable link request (QOLELINK). If the enable link request is not successful (return and reason codes are nonzero), the user spaces are not created. In all, there are four user spaces that are created. Two of these are used for input and two for output. In each set of two spaces, one is used as a buffer to contain user data, and the other is used as a descriptor, to describe the data (length and any other qualifiers to the data).

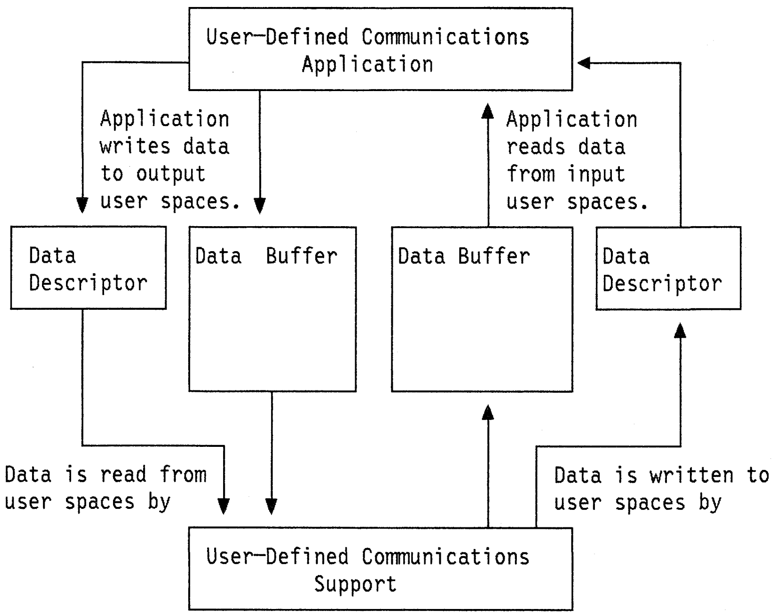

User-Defined Communications Application
output
input
Data Descriptor
Data Buffer
Data Buffer
Data Descriptor
User-Defined Communications Support

The application provides the library and name of the user space object that is to be created. The descriptive text of the object always contains the name of the job that is using these spaces. Finally, when the link is disabled (either explicitly or implicitly) these user spaces will be deleted by the user-defined communications support. Refer to "QOLDLINK" on page 2-8 for information on disabling the link.

The application writes to the output buffer and descriptor and reads from the input buffer and descriptor. Similarly, the communications support reads from the output buffer and descriptor and writes to the input buffer and descriptor. As soon as the call to QOLSEND or QOLRECV has completed, the application can access these user spaces.

Note: Since the user spaces are system objects, appropriate security should be enforced. An object will inherit security from the job attributes of the application and the library that the object is created in. Using library QTEMP is one way to ensure that only the user-defined communications application has access to the user spaces.

If these user spaces are changed or deleted while in use by the user-defined communications support, a severe application error will be reported to the application and communications over the link associated with the user spaces will no longer be possible.


User-Defined Communications Application
Application writes data to output user spaces.
Application reads data from input user spaces.
Data Descriptor
Data Buffer
Data Buffer
Data Descriptor
Data is read from user spaces by
Data is written to user spaces by
User-Defined Communications Support

These spaces have logical views defined by user-defined communications support. These views are sometimes referred to as formats. There is a format for filters, sending and receiving LAN frames, and sending and receiving X.25 packets. See "QOLSEND" on page 2-17 and "QOLRECV" on page 2-40 for details on these formats.

The application is responsible for setting all the data required for the format. There are two types of byte fields in the buffer and descriptors, character (CHAR) and binary (BIN). Binary implies that the value will be used as a numeric value. Sometimes this might be a one-byte numeric value; for example, 12 = X'0C'. If the language the application is written in is not capable of setting this type of binary field, the field should be declared as character and set to X'0C'. The character type contains an EBCDIC value, printable or unprintable. All parameter values are either character or 4-byte binary. See "Programming Languages" on page 3-5 for help in converting between the language your application is written in and the expected input by the user-defined communications support.

The communications support will never change the output buffer, therefore, the application is responsible for initializing the buffer and descriptor for the next operation to use if necessary. The data in the output buffer can also be used to help determine why a particular operation is not successful.

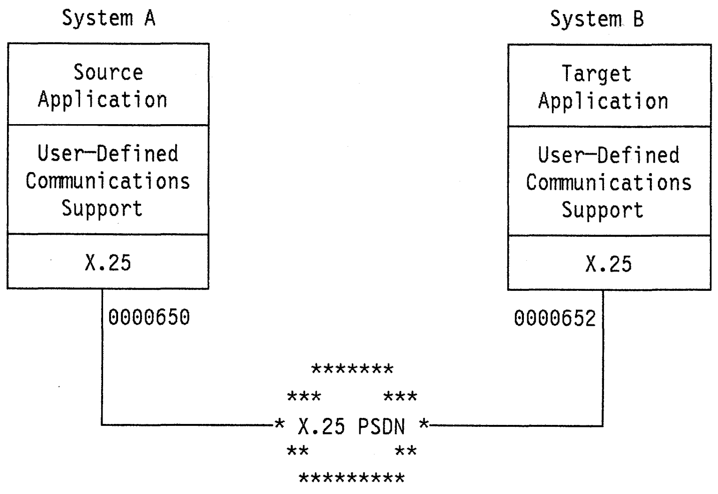
For performance reasons, the application should attempt to fill the output buffer as completely as possible.

Finally, for security reasons, the application will choose the library the user space object will reside in. The library can be any system library, including QTEMP. The advantage (or disadvantage) of using QTEMP for user space objects is that only the job which has enabled the links has access to the user spaces. This is because a QTEMP library exists for each job on the system. If the user space objects are in any other library, any job having authority to the library the user spaces are in can access them.

Chapter 4. Application Programming Examples

The purpose of this section is to provide a simple example of how X.25-oriented applications use the user-defined communications support to connect to remote systems. Two user-defined application examples written in the C programming language will be used to illustrate a simple file transfer between two systems over an X.25 packet switched data network (PSDN). An important note to remember is that although an X.25 example is shown, many of the same concepts can be applied to applications running over token-ring and Ethernet local area networks (LANs).

For this example, the following network configuration will be used.



X.25 Overview

In this example X.25 network, the source application is responsible for establishing a switched virtual circuit, or connection to the target application running on System B. This is accomplished by using the remote network address (System B's address) of X'0000652'. After the target application on System B is initialized, it will wait for notification of an incoming call packet before proceeding. Once the virtual circuit is established, the source application reads records from a file into its output buffer and sends them to the target application using normal X.25 data transfer procedures. When receiving the file data, the target application writes the data to a local file on System B. When the file transfer completes, the source application closes the connection by issuing an X.25 clear request packet and ends. When receiving the clear indication packet, the target application also ends.

User-Defined Communications Support Overview

Both the source and target applications call the QOLQLIND API to obtain information about the local X.25 line being used. This information is stored in a local control block for use in establishing the peer connection during X.25 connection processing. Both applications also call the QOLELINK API to enable the link for future communications. The AS/400 line name, communications handle, and remote DTE address are passed to both programs as arguments to the C function main(). For simplicity, the user space names and data queue name on the call to the QOLELINK API are hard coded directly in the applications.

Note: Keyed data queue support is used by both applications. The key length is 6 and the keys used are Source and Target for the source and target applications, respectively.

Activating Filters

Once the links have been enabled and both applications have read their respective enable-complete entries from their data queues, the target application program calls the QOLSETF API to activate a filter. The filter activated then identifies the protocol of the local X.25 service user. This filter is used by the user-defined communications support on System B to route incoming calls. The actual filter type activated is X'00' (for X.25 PID) and its associated value is X'21'. For more information concerning filters, see "QOLSETF" on page 2-10. After activating the X'21' filter, the target application will wait for the source application to request a connection.

Establishing a Connection

The source application will call the QOLSEND API with a X'B000' operation in its output data buffer to establish an SVC to the target application. Included in the first byte of the call user data is the protocol ID of the target application, or X'21'. When the user-defined communications support on System B sees the incoming call packet with the first byte of user data equal to a previously activated filter, the call is routed to the process responsible for activating that filter. In this case, the target application will receive notification of an incoming call since it previously activated filter X'21'.

The target application, waiting for the incoming call by design, calls the QOLRECV API to receive a X'B201' operation with incoming call data. After doing so, the target application will accept the X.25 connection by calling the QOLSEND API with a X'B400' operation in its output data buffer. See "QOLSEND" on page 2-17 for more information.

Sending Data

As mentioned previously, once the peer connection has been established between the source and target applications running on System A and System B respectively, the file transfer takes place. The source application reads records from a local file and calls the QOLSEND API with X'0000' operations in its output data buffer to transfer the file data to System B. This process continues until the entire contents of the source file has been sent to System B.

Receiving Data

After accepting the X.25 connection, the target application waits until its data queue receives incoming-data entries. When one is read from the queue, the QOLRECV API is called to determine what operation was received. Barring failure, the target application should receive a X'0001' operation as a result of the QOLRECV API call. The data contained in the input data buffer will be the file data received from System A. When receiving the file data, the target application will write the data to a local file. This process continues until the entire contents of the file is received from System A. By design, the target application assumes the file transfer is complete when an operation other than a X'0001' operation is received after a successful call to the QOLRECV API. Most likely, the first non-X'0001' operation received will be X'B301' operation, signalling that the user-defined communications support running on System B received an SVC clear indication.

Clearing the Connection and Disabling Links

Once the entire contents of the file has been read and sent to System B, the source application calls the QOLSEND API with a X′B100′ operation in its output data buffer to clear the X.25 connection. Afterwards, the source application closes its local file, disables its local link by calling the QOLDLINK API, and ends.

When the source application program sends a X′B100′ operation, it causes the target application to receive a X′B301′ operation. When receiving this operation, the target application program will call the QOLSEND API with a X′B100′ operation to locally close the connection between itself and the user-defined communications support. Afterwards, the target application closes its local file, disables its local link by calling the QOLDLINK API, and ends.

Using Timers and the Data Queue Support

Both the source and target application programs use the user-defined communications support timer service to manage the reception of certain operations. This is accomplished by setting a timer before checking the data queue for an entry. For example, the target application sets a timer to manage the reception of file data from the source application. If the timer expires, the user-defined communications support will place a timer-expired entry on the application's data queue. By design the target application assumes when receiving this entry that the source application ended abnormally and it will, therefore, take appropriate action to end itself.

C/400 Compiler Listings

Below are listings of the source and target applications described in the previous paragraphs. Note the block numbers in the listings. Detailed explanations of each block will follow.

Source Application on System A Listing

As mentioned above, the source application in this example is the initiator of all meaningful work. In other words, the source program listed on the following pages performs the following:

- Calls the QOLQLIND API to get local X.25 line information
- Opens the local file
- Calls the QOLELINK API to establish a link for communications
- Calls the QOLSEND API with X′B000′ operation to establish a peer (SVC) connection
- Sends the local file to the target system via X′0000′ operations
- Calls the QOLSEND API with X′B100′ operation to clear the peer (SVC) connection
- Calls the QOLDLINK API to disable the link
- Calls the QOLTIMER API to manage the reception of data queue entries

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                 S1016897        12/19/90 08:51:20      Page      1
                                              * * * * *   P R O L O G   * * * * *
 Program name . . . . . . . . . . :   SOURCE
   Library name . . . . . . . . . :      UDCS_APPLS
 Source file . . . . . . . . . . . :   QCSRC
   Library name . . . . . . . . . :      UDCS_APPLS
 Source member name . . . . . . . :   SOURCE
 Text Description . . . . . . . . :   Source Application Example
 Compiler options . . . . . . . . :   *SOURCE     *NOXREF     *SHOWUSR    *SHOWSYS    *NOSHOWSKP  *NOEXPMAC   *NOAGR
                                  :   *NOPPONLY   *NODEBUG    *GEN        *NOSECLVL   *PRINT      *LOGMSG
 Language level options . . . . . :
 Source margins:
   Left margin . . . . . . . . . . :   1
   Right margin . . . . . . . . . :   80
 Sequence columns:
   Left Column . . . . . . . . . . :
   Right Column . . . . . . . . . :
 Define name . . . . . . . . . . . :
 Generation options . . . . . . . :   *NOLIST     *NOXREF     *GEN        *NOATR      *NODUMP     *NOOPTIMIZE *NOALWBND
                                  :   *NOANNO
 Print file . . . . . . . . . . . :   QSYSPRT
   Library name . . . . . . . . . :      *LIBL
 Message flagging level . . . . . :   0
 Compiler message:
   Message limit . . . . . . . . . :   *NOMAX
   Message limit severity . . . . :   30
 Replace program object . . . . . :   *YES
 User profile . . . . . . . . . . :   *USER
 Authority . . . . . . . . . . . . :   *LIBCRTAUT
 Target Release . . . . . . . . . :   *CURRENT
 INDEBUG options . . . . . . . . . :   I don't know
 Last change . . . . . . . . . . . :   90/12/19 08:49:37
 Source description . . . . . . . :   Source Application Example
 Compiler . . . . . . . . . . . . :   IBM SAA C/400 Compiler
```

Figure 4-1 (Part 1 of 26). C/400 Compiler Listing for the Source Application

```
                                          * * * * *   S O U R C E   * * * * *
 Line  STMT                                                                                                    SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
         |/*******************************************************************/                          |
         |/**                                                             **/                          |
         |/**  Program Name: Source Application Program Example           **/                          |
         |/**                                                             **/                          |
         |/**                                                             **/                          |
         |/**  Function:                                                  **/                          |
         |/**  This is the source application program example that uses   **/                          |
         |/**  X.25 services provided by the user-defined communications  **/                          |
         |/**  support to transfer a simple file to the target application **/                         |
         |/**  program running on system B.  This program performs the    **/                          |
         |/**  following:                                                 **/                          |
         |/**     01.  Open the source file name INFILE.                  **/                          |
         |/**     02.  Call QOLQLIND API to obtain local line information. **/                         |
         |/**     03.  Enable a link.                                     **/                          |
         |/**     04.  Send a 'B000'X operation (call request).           **/                          |
         |/**     05.  Receive a 'B001'X operation (call confirmation).   **/                          |
         |/**     06.  Read record(s) from the file opened in step 1). and **/                         |
         |/**          send '0001'X operation(s) to transfer the file to  **/                          |
         |/**          the target application program.                    **/                          |
         |/**     07.  Send a 'B100'X operation (clear call request).     **/                          |
         |/**     08.  Receive a 'B101'X operation.                       **/                          |
         |/**     09.  Disable the link enabled in step 3).               **/                          |
         |/**                                                             **/                          |
         |/**  A data queue will be actively used to manage the operation **/                          |
         |/**  of this program.  Data queue support will be used to monitor **/                        |
         |/**  for the completion of the enable and disable routines, as  **/                          |
         |/**  well as timer expirations and incoming data.  Timers are   **/                          |
         |/**  used to ensure that there will never be an infinite wait on **/                         |
         |/**  the data queue.  If a timer expires, the link enabled will **/                          |
         |/**  be disabled and the program will stop.                     **/                          |
         |/**                                                             **/                          |
         |/**                                                             **/                          |
         |/** Inputs:                                                     **/                          |
         |/** The program expects the following input parameters          **/                          |
         |/**      Line Name:   This is the name of the line description  **/                          |
         |/**                   that will be used to call the QOLELINK API. **/                        |
         |/**                   The line must be an X.25 line with at least **/                        |
         |/**                   one SVC of type *SVCBOTH or *SVCOUT.      **/                          |
         |/**                                                             **/                          |
         |/**      CommHandle:  This is the logical name that will be used **/                         |
         |/**                   to identify the link enabled.             **/                          |
         |/**                                                             **/                          |
         |/**      Remote DTE Address:  The is the Local Network Address  **/                          |
         |/**                           of System B.                      **/                          |
         |/**                                                             **/                          |
         |/**                                                             **/                          |
         |/** Outputs:                                                    **/                          |
         |/** Current status of the file transfer will be provided when   **/                          |
         |/** running this program.  If an error should occur, then a     **/                          |
         |/** message will be displayed indicating where the error occurred **/                        |
         |/** and the program will end.        If the program completes   **/                          |
         |/** successfully, a "successful completion" message will be     **/                          |
```

Figure 4-1 (Part 2 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                S1016897        12/19/90 08:51:20      Page    3
 Line  STMT                                                                                                       SEQNBR  INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
   53        |/** posted.                                                      **/                       |     53
   54        |/**                                                              **/                       |     54
   55        |/*****************************************************************/                       |     55
   56        |                                                                                           |     56
   57        |#include "header"                                                                          |     57
    1        |                                                                                           |     58       1
    2        |#include "typedefs"                                                                        |     59       1
    1        |                                                                                           |     58       2
    2        |                                                                                           |     58       2
    3        |                                                                                           |     62       2
    4        |#include <stdio.h>                                                                         |     63       2
    5        |#include <stdlib.h>                                                                        |    676       2
    6        |#include <signal.h>                                                                        |    798       2
    7        |#include <xxasio.h>                                                                        |    922       2
    8        |#include <xxcvt.h>                                                                         |    970       2
    9        |#include <string.h>                                                                        |    999       2
   10        |#include <ctype.h>                                                                         |   1095       2
              1
   11        |                                                                                           |   1192       2
   12        |typedef struct queuein                                                                     |   1193       2
   13        |    {                                                                                      |   1194       2
   14        |      char library??(10??);                                                                |   1195       2
   15        |      char name??(10??);                                                                   |   1196       2
   16        |      char option;                                                                         |   1197       2
   17        |    } queuein;                                                                             |   1198       2
   18        |                                                                                           |   1199       2
   19        |typedef struct namelib                                                                     |   1200       2
   20        |    {                                                                                      |   1201       2
   21        |      char library??(10??);                                                                |   1202       2
   22        |      char name??(10??);                                                                   |   1203       2
   23        |    } namelib;                                                                             |   1204       2
   24        |                                                                                           |   1205       2
   25        |typedef _Packed struct format1                                                             |   1206       2
   26        |  {                                                                                        |   1207       2
   27        |   char type;                                                                              |   1208       2
   28        |   char reserved1;                                                                         |   1209       2
   29        |   unsigned short logchanid;                                                               |   1210       2
   30        |   unsigned short sendpacksize;                                                            |   1211       2
   31        |   unsigned short sendwindsize;                                                            |   1212       2
   32        |   unsigned short recvpacksize;                                                            |   1213       2
   33        |   unsigned short recvwindsize;                                                            |   1214       2
   34        |   char reserved2??(7??);                                                                  |   1215       2
   35        |   char dtelength;                                                                         |   1216       2
   36        |   char dte??(16??);                                                                       |   1217       2
   37        |   char reserved3??(8??);                                                                  |   1218       2
   38        |   char dbit;                                                                              |   1219       2
```

Figure 4-1 (Part 3 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE              S1016897       12/19/90 08:51:20     Page    21
 Line  STMT                                                                                                   SEQNBR   INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
   39       |   char reserved4??(7??);                                                                  |    1220       2
   40       |   char cug;                                                                               |    1221       2
   41       |   char cugid;                                                                             |    1222       2
   42       |   char reverse;                                                                           |    1223       2
   43       |   char fast;                                                                              |    1224       2
   44       |   char faclength;                                                                         |    1225       2
   45       |   char facilities??(109??);                                                               |    1226       2
   46       |   char reserved5??(48??);                                                                 |    1227       2
   47       |   unsigned short calllength;                                                              |    1228       2
   48       |   char calllud??(128??);                                                                  |    1229       2
   49       |   char reserved6??(128??);                                                                |    1230       2
   50       |   unsigned char misc??(4??);          /* control flags */                                 |    1231       2
   51       |   unsigned int maxasmsize;                                                                |    1232       2
   52       |   unsigned short autoflow;                                                                |    1233       2
   53       |} format1;                                                                                 |    1234       2
   54       |                                                                                           |    1235       2
   55       |typedef _Packed struct format2                                                             |    1236       2
   56       |  {                                                                                        |    1237       2
   57       |   unsigned short type;                                                                    |    1238       2
   58       |   char cause;                                                                             |    1239       2
   59       |   char diagnostic;                                                                        |    1240       2
   60       |   char reserved??(4??);                                                                   |    1241       2
   61       |   char faclength;                                                                         |    1242       2
   62       |   char facilities??(109??);                                                               |    1243       2
   63       |   char reserved2??(48??);                                                                 |    1244       2
   64       |   unsigned short length;                                                                  |    1245       2
   65       |   char userdata??(128??);                                                                 |    1246       2
   66       |  } format2;                                                                               |    1247       2
   67       |                                                                                           |    1248       2
   68       |typedef _Packed struct desc                                                                |    1249       2
   69       |  {                                                                                        |    1250       2
   70       |   unsigned short length;                                                                  |    1251       2
   71       |   char more;              /*These 4 char's are only used for X.25.*/                      |    1252       2
   72       |   char qualified;                                                                         |    1253       2
   73       |   char interrupt;                                                                         |    1254       2
   74       |   char dbit;                                                                              |    1255       2
   75       |   char reserved??(26??);                                                                  |    1256       2
   76       |  } desc;                                                                                  |    1257       2
   77       |                                                                                           |    1258       2
   78       |typedef _Packed struct llcheader                                                           |    1259       2
   79       |  {                                                                                        |    1260       2
   80       |   unsigned short headerlength;                                                            |    1261       2
   81       |   char macaddr??(6??);                                                                    |    1262       2
   82       |   char dsap;                                                                              |    1263       2
   83       |   char ssap;                                                                              |    1264       2
   84       |   char priority;                                                                          |    1265       2
   85       |   char priorctl;                                                                          |    1266       2
   86       |   unsigned short routlen;                                                                 |    1267       2
   87       |   unsigned short userdtalen;                                                              |    1268       2
   88       |   char data??(1??);                                                                       |    1269       2
   89       |  } llcheader;                                                                             |    1270       2
   90       |                                                                                           |    1271       2
   91       |typedef _Packed struct espec                                                               |    1272       2
   92       |  {                                                                                        |    1273       2
```

Figure 4-1 (Part 4 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                 S1016897        12/19/90 08:51:20     Page    22
 Line  STMT                                                                                                         SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
   93       |   char reserved??(2??);                                                                        |   1274        2
   94       |   unsigned int hwecode;                                                                        |   1275        2
   95       |   unsigned int timestamphi;                                                                    |   1276        2
   96       |   unsigned int timestamplo;                                                                    |   1277        2
   97       |   unsigned int elogid;                                                                         |   1278        2
   98       |   char reserved2??(10??);                                                                      |   1279        2
   99       |   char flags;                                                                                  |   1280        2
  100       |   char cause;                                                                                  |   1281        2
  101       |   char diagnostic;                                                                             |   1282        2
  102       |   char reserved3;                                                                              |   1283        2
  103       |   unsigned int erroroffset;                                                                    |   1284        2
  104       |   char reserved4??(4??);                                                                       |   1285        2
  105       |  } espec;                                                                                      |   1286        2
  106       |                                                                                                |   1287        2
  107       |typedef struct tableentry                                                                       |   1288        2
  108       |  {                                                                                             |   1289        2
  109       |    char handle??(10??);                                                                        |   1290        2
  110       |    char type;                                                                                  |   1291        2
  111       |    char inbuff??(20??);                                                                        |   1292        2
  112       |    char indesc??(20??);                                                                        |   1293        2
  113       |    char outbuff??(20??);                                                                       |   1294        2
  114       |    char outdesc??(20??);                                                                       |   1295        2
  115       |    unsigned int totaldusize;                                                                   |   1296        2
  116       |    struct tableentry *next;                                                                    |   1297        2
  117       |  } tableentry;                                                                                 |   1298        2
  118       |                                                                                                |   1299        2
  119       |/*******  Data structure for X.25 line      ********/                                           |   1300        2
  120       |/*******  descriptions as returned by QOLQLIND.  ******/                                        |   1301        2
  121       |                                                                                                |   1302        2
  122       |typedef struct x25info                                                                          |   1303        2
  123       |  {                                                                                             |   1304        2
  124       |   char addrlen;                                                                                |   1305        2
  125       |   char addr??(9??);                                                                            |   1306        2
  126       |   char addrtype;                                                                               |   1307        2
  127       |   char insert;                                                                                 |   1308        2
  128       |   char modulus;                                                                                |   1309        2
  129       |   char dtedce;                                                                                 |   1310        2
  130       |   unsigned short maxsend;                                                                      |   1311        2
  131       |   unsigned short maxrecv;                                                                      |   1312        2
  132       |   unsigned short defsend;                                                                      |   1313        2
  133       |   unsigned short defrecv;                                                                      |   1314        2
  134       |   char windowsend;                                                                             |   1315        2
  135       |   char windowrecv;                                                                             |   1316        2
  136       |   unsigned short numlc;                                                                        |   1317        2
  137       |   char lcinfo??(4??);                                                                          |   1318        2
  138       |  } x25info;                                                                                    |   1319        2
  139       |                                                                                                |   1320        2
  140       |typedef struct querydata                                                                        |   1321        2
  141       |  {                                                                                             |   1322        2
  142       |   char header??(12??);     /* line header info */                                              |   1323        2
  143       |   x25info x25data;         /* preliminary data */                                              |   1324        2
  144       |  } querydata;                                                                                  |   1325        2
    3       |#include "hexconv"                                                                              |   1326        1
    1       |#include <stdio.h>                                                                              |   1327       15
```

Figure 4-1 (Part 5 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE               S1016897        12/19/90 08:51:20     Page    23
 Line  STMT                                                                                                          SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
   2          |                                                                                                 |  1532      15
   3          |char *inttohex(decimal,hex)  /*Converts a 4-byte integer into a                                  |  1533      15
   4          |                               string of 2 uppercase hex characters.*/                           |  1534      15
   5          |unsigned int decimal;                                                                            |  1535      15
   6          |char *hex;                                                                                       |  1536      15
   7          |                                                                                                 |  1537      15
   8          |{                                                                                                |  1538      15
   9       1  |  sprintf(hex,"%.2X",decimal);                                                                   |  1539      15
  10       2  |  return(hex);                                                                                   |  1540      15
  11          |}                                                                                                |  1541      15
  12          |                                                                                                 |  1542      15
  13          |unsigned int hextoint(hex)   /*Converts a string containing hex                                  |  1543      15
  14          |                               digits into a 4-byte integer.   */                                |  1544      15
  15          |char *hex;                                                                                       |  1545      15
  16          |{                                                                                                |  1546      15
  17          |  int decimal;                                                                                   |  1547      15
  18          |                                                                                                 |  1548      15
  19       1  |  sscanf(hex,"%x",&decimal);                                                                     |  1549      15
  20       2  |  return(decimal);                                                                               |  1550      15
  21          |}                                                                                                |  1551      15
               2
   4          |                                                                                                 |  1552       1
   5          |#pragma linkage(QOLDLINK, OS)                                                                    |  1553       1
   6          |#pragma linkage(QOLELINK, OS)                                                                    |  1554       1
   7          |#pragma linkage(QOLSEND, OS)                                                                     |  1555       1
   8          |#pragma linkage(QOLRECV, OS)                                                                     |  1556       1
   9          |#pragma linkage(QUSPTRUS, OS)                                                                    |  1557       1
  10          |#pragma linkage(QRCVDTAQ, OS)                                                                    |  1558       1
  11          |#pragma linkage(QCLRDTAQ, OS)                                                                    |  1559       1
  12          |#pragma linkage(QOLTIMER, OS)                                                                    |  1560       1
  13          |#pragma linkage(QOLSETF, OS)                                                                     |  1561       1
  14          |#pragma linkage(QOLQLIND, OS)                                                                    |  1562       1
  15          |                                                                                                 |  1563       1
  16          |FILE *screen;                                                                                    |  1564       1
  17          |FILE *rptr;                                                                                      |  1565       1
  18          |FILE *fptr;                                                                                      |  1566       1
  19          |                                                                                                 |  1567       1
```

Figure 4-1 (Part 6 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE              S1016897       12/19/90 08:51:20     Page    24
 Line  STMT                                                                                                    SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
   20       |extern void QOLDLINK(int *, int *, char *, char *);                                          |   1568        1
             3
   21       |                                                                                             |   1569        1
   22       |extern void QOLELINK (int *, int *, int *, int *, int *, int *,\                             |   1570        1
   23       |                      char *, char *, char *, char *, int *, char *,\                        |   1571        1
   24       |                      char *, char *, char *);                                               |   1572        1
   25       |                                                                                             |   1573        1
   26       |extern void QOLSEND (int *, int *, void *, int *, int *, int *,\                             |   1574        1
   27       |                      char *, unsigned short *, int *);                                      |   1575        1
   28       |                                                                                             |   1576        1
   29       |extern void QOLRECV (int *, int *, int *, int *, unsigned short *,\                          |   1577        1
   30       |                      int *, char *, void *, char *);                                        |   1578        1
   31       |                                                                                             |   1579        1
   32       |extern void QOLSETF (int *, int *, int *, char *);                                           |   1580        1
   33       |                                                                                             |   1581        1
   34       |extern void QOLTIMER (int *, int *, char *, char *, char *, char *,\                         |   1582        1
   35       |                      int *, int *, int *, char *, char *);                                  |   1583        1
   36       |                                                                                             |   1584        1
   37       |extern void QUSPTRUS (void *, void *);                                                       |   1585        1
   38       |                                                                                             |   1586        1
   39       |extern void QRCVDTAQ (char *, char *, char *, void *, char *,\                               |   1587        1
   40       |                      char *, char *, char *, char *, char *);                               |   1588        1
   41       |                                                                                             |   1589        1
   42       |extern void QCLRDTAQ (char *, char *);                                                       |   1590        1
   43       |                                                                                             |   1591        1
   44       |extern void QOLQLIND(int *, int *, int *, void *, char *, char *);                           |   1592        1
   45       |                                                                                             |   1593        1
   46       |/************   Typedef  Declarations      *******************/                             |   1594        1
   47       |                                                                                             |   1595        1
   48       |typedef struct usrspace                                                                      |   1596        1
   49       |    {                                                                                        |   1597        1
   50       |      char name??(10??);                                                                     |   1598        1
   51       |      char library??(10??);                                                                  |   1599        1
   52       |    } usrspace;                                                                              |   1600        1
             4
   53       |                                                                                             |   1601        1
   54       |typedef struct enableparms     /*  Enable parameters  */                                     |   1602        1
   55       |  {                                                                                          |   1603        1
   56       |   int retcode,            /*  Output  */                                                    |   1604        1
   57       |       reason,             /*  Output  */                                                    |   1605        1
   58       |       tdusize,            /*  Output  */                                                    |   1606        1
   59       |       numunits,           /*  Output  */                                                    |   1607        1
   60       |       maxdtalan,          /*  Output  */                                                    |   1608        1
   61       |       maxdtax25,          /*  Input   */                                                    |   1609        1
   62       |       keylength;          /*  Input   */                                                    |   1610        1
   63       |char keyvalue??(256??),     /*  Input  */                                                    |   1611        1
   64       |     linename??(10??);      /*  Input  */                                                    |   1612        1
   65       |  } enableparms;                                                                             |   1613        1
   66       |                                                                                             |   1614        1
   67       |typedef struct disableparms     /*  Disable parameters  */                                   |   1615        1
   68       |  {                                                                                          |   1616        1
   69       |  int retcode,             /*  Output  */                                                    |   1617        1
   70       |      reason;              /*  Output  */                                                    |   1618        1
   71       |  char vary;               /*  Input  */                                                     |   1619        1
   72       |  } disableparms;                                                                            |   1620        1
   73       |                                                                                             |   1621        1
```

Figure 4-1 (Part 7 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                 S1016897        12/19/90 08:51:20      Page    25
 Line  STMT                                                                                                           SEQNBR   INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
   74       |typedef struct setfparms     /*  Set Filters parameters  */                                        |   1622        1
   75       |  {                                                                                                 |   1623        1
   76       |  int retcode,            /*  Output  */                                                            |   1624        1
   77       |      reason,             /*  Output  */                                                            |   1625        1
   78       |      erroffset;          /*  Output  */                                                            |   1626        1
   79       |  } setfparms;                                                                                      |   1627        1
   80       |                                                                                                    |   1628        1
   81       |typedef _Packed struct hdrparms /* Filter header */                                                 |   1629        1
   82       |  {                                                                                                 |   1630        1
   83       |   char function;                                                                                   |   1631        1
   84       |   char type;                                                                                       |   1632        1
   85       |   unsigned short number;                                                                           |   1633        1
   86       |   unsigned short length;                                                                           |   1634        1
   87       |   char filters??(1??);                                                                             |   1635        1
   88       |  } hdrparms;                                                                                       |   1636        1
   89       |                                                                                                    |   1637        1
   90       |typedef _Packed struct x25filter /* X.25 filter */                                                  |   1638        1
   91       |  {                                                                                                 |   1639        1
   92       |   char pidlength;                                                                                  |   1640        1
   93       |   char pid;                                                                                        |   1641        1
   94       |   char dtelength;                                                                                  |   1642        1
   95       |   char dte??(12??);                                                                                |   1643        1
   96       |   char flags;                                                                                      |   1644        1
   97       |  } x25filter;                                                                                      |   1645        1
   98       |                                                                                                    |   1646        1
   99       |typedef struct sendparms     /*  Send parameters  */                                                |   1647        1
  100       |  {                                                                                                 |   1648        1
  101       | espec errorspecific;      /*  Output  */                                                         |   1649        1
  102       |   int retcode,            /*  Output  */                                                         |   1650        1
  103       |       reason,             /*  Output  */                                                         |   1651        1
  104       |       newpcep,            /*  Output  */                                                         |   1652        1
  105       |       ucep,               /*  Input  */                                                          |   1653        1
  106       |       pcep,               /*  Input   */                                                         |   1654        1
  107       |       numdtaelmnts;       /*  Input   */                                                         |   1655        1
  108       |unsigned short operation;  /*  Input  */                                                          |   1656        1
  109       |  } sendparms;                                                                                      |   1657        1
  110       |                                                                                                    |   1658        1
  111       |typedef struct recvparms     /*  Receive parameters  */                                             |   1659        1
  112       |  {                                                                                                 |   1660        1
  113       | espec errorspecific;      /*  Output  */                                                         |   1661        1
  114       |   int retcode,            /*  Output  */                                                         |   1662        1
  115       |       reason,             /*  Output  */                                                         |   1663        1
  116       |       newpcep,            /*  Output  */                                                         |   1664        1
  117       |       ucep,               /*  Output  */                                                         |   1665        1
  118       |       pcep,               /*  Output   */                                                        |   1666        1
  119       |       numdtaunits;        /*  Output   */                                                        |   1667        1
  120       |  char dataavail;          /*  Output  */                                                         |   1668        1
  121       |unsigned short operation;  /*  Output  */                                                         |   1669        1
  122       |  } recvparms;                                                                                      |   1670        1
  123       |                                                                                                    |   1671        1
  124       |typedef struct timerparms     /*  Timer parameters  */                                              |   1672        1
  125       |  {                                                                                                 |   1673        1
  126       |   int retcode,            /*  Output  */                                                         |   1674        1
  127       |       reason,             /*  Output  */                                                         |   1675        1
```

Figure 4-1 (Part 8 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE               S1016897        12/19/90 08:51:20      Page    26
 Line  STMT                                                                                                          SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  128       |       interval,           /*  Input  */                                                          |   1676         1
  129       |       establishcount,     /*  Input  */                                                          |   1677         1
  130       |       keylength;          /*  Input   */                                                         |   1678         1
  131       |  char handleout??(8??),   /*  Output  */                                                         |   1679         1
  132       |       handlein??(8??),    /*  Input  */                                                          |   1680         1
  133       |       operation,          /*  Input  */                                                          |   1681         1
  134       |       keyvalue??(256??),  /*  Input  */                                                          |   1682         1
  135       |       userdata??(60??);   /*  Input  */                                                          |   1683         1
  136       |  } timerparms;                                                                                   |   1684         1
  137       |                                                                                                  |   1685         1
  138       |                                                                                                  |   1686         1
  139       |typedef struct especout                                                                           |   1687         1
  140       |  {                                                                                               |   1688         1
  141       |   char hwecode??(8??);                                                                           |   1689         1
  142       |   char timestamp??(16??);                                                                        |   1690         1
  143       |   char elogid??(8??);                                                                            |   1691         1
  144       |   char fail;                                                                                     |   1692         1
  145       |   char zerocodes;                                                                                |   1693         1
  146       |   char qsysopr;                                                                                  |   1694         1
  147       |   char cause??(2??);                                                                             |   1695         1
  148       |   char diagnostic??(2??);                                                                        |   1696         1
  149       |   char erroffset??(6??);                                                                         |   1697         1
  150       |  } especout;                                                                                     |   1698         1
  151       |                                                                                                  |   1699         1
  152       |typedef struct qlindparms     /*  Query line parameters  */                                       |   1700         1
  153       |  {                                                                                               |   1701         1
  154       |   int retcode,              /*  Output  */                                                       |   1702         1
  155       |       reason,               /*  Output  */                                                       |   1703         1
  156       |       nbytes;               /*  Output  */                                                       |   1704         1
  157       |   char userbuffer??(256??);                                                                      |   1705         1
  158       |   char format;                                                                                   |   1706         1
  159       |  } qlindparms;                                                                                   |   1707         1
  160       |                                                                                                  |   1708         1
  161       |typedef _Packed union content       /*  Queue support parameters  */                              |   1709         1
  162       |  {                                                                                               |   1710         1
  163       |    _Packed struct other                                                                          |   1711         1
  164       |      {                                                                                           |   1712         1
  165       |       char commhandle??(10??);                                                                   |   1713         1
  166       |       char reserved??(58??);                                                                     |   1714         1
  167       |      } other;                                                                                    |   1715         1
  168       |    _Packed struct enable                                                                         |   1716         1
  169       |      {                                                                                           |   1717         1
  170       |       char commhandle??(10??);                                                                   |   1718         1
  171       |       char status;                                                                               |   1719         1
  172       |       char reserved??(57??);                                                                     |   1720         1
  173       |      } enable;                                                                                   |   1721         1
  174       |    _Packed struct timer                                                                          |   1722         1
  175       |      {                                                                                           |   1723         1
  176       |       char timerhandle??(8??);                                                                   |   1724         1
  177       |       char userdata??(60??);                                                                     |   1725         1
  178       |      } timer;                                                                                    |   1726         1
  179       |  } content;                                                                                      |   1727         1
  180       |                                                                                                  |   1728         1
  181       |typedef _Packed struct qentry      /*  Queue  parameters  */                                      |   1729         1
```

Figure 4-1 (Part 9 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                  S1016897        12/19/90 08:51:20     Page    27
 Line  STMT                                                                                                           SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  182       | {                                                                                              |  1730       1
  183       |   char type??(10??);                                                                           |  1731       1
  184       |   unsigned short msgid;                                                                        |  1732       1
  185       |   content message;                                                                             |  1733       1
  186       |   char key??(256??);                                                                           |  1734       1
  187       | } qentry;                                                                                      |  1735       1
  188       |                                                                                                |  1736       1
             5
   58       |                                                                                                |  1737
   59       |void senddata(sendparms *a, char *b, desc *c, char *d, char *e, int f);                         |  1738
   60       |                                                                                                |  1739
   61       |void sndformat1(sendparms *a,char *b, char *c, char *d, qlindparms *f);                         |  1740
   62       |                                                                                                |  1741
   63       |void sndformat2 (sendparms *a, char *b, char *c);                                               |  1742
   64       |                                                                                                |  1743
   65       |void setfilters (hdrparms *a);                                                                  |  1744
   66       |                                                                                                |  1745
   67       |void byte (char *a, int b, char *c, int d);                                                     |  1746
   68       |                                                                                                |  1747
   69       |void printespec (espec *a);                                                                     |  1748
   70       |                                                                                                |  1749
   71       |void settimer(unsigned short *a,char *b,qentry *c,usrspace *d,char *e);                         |  1750
   72       |                                                                                                |  1751
   73       |void dequeue (int a, char *b, qentry *c, usrspace *d);                                          |  1752
   74       |                                                                                                |  1753
   75       |void x25lind (qlindparms *a, char *b);                                                          |  1754
   76       |                                                                                                |  1755
   77       |int getline (char *a, int b, FILE *c);                                                          |  1756
   78       |                                                                                                |  1757
   79       |void disablelink (disableparms *a, char *b, usrspace *c);                                       |  1758
   80       |                                                                                                |  1759
   81       |void handler (disableparms a, usrspace *b);                                                     |  1760
   82       |                                                                                                |  1761
   83       |sigdata_t *sigdata(void);                                                                       |  1762
   84       |                                                                                                |  1763
   85       |/**********************************************************/                                    |  1764
   86       |/***************   Start Main Program   ******************/                                    |  1765
   87       |/**********************************************************/                                    |  1766
   88       |                                                                                                |  1767
   89       |main (int argc, char *argv??(??))                                                               |  1768
   90       |{                                                                                               |  1769
   91       |                                                                                                |  1770
   92       |/************  Variable  Declarations     *******************/                                  |  1771
   93       |                                                                                                |  1772
   94       |  usrspace inbuff,       /*  Input Data Buffer */                                              |  1773
   95       |           indesc,       /*  Input Buffer Descriptor  */                                       |  1774
   96       |           outbuff,      /*  Output Data Buffer  */                                            |  1775
   97       |           outdesc,      /*  Output Buffer Descriptor */                                       |  1776
   98       |           qname;        /*  Data Queue  */                                                    |  1777
   99       |                                                                                                |  1778
  100       |  int length,                 /* Data Queue key legth  */                                     |  1779
  101       |      linesiz,                /* Length of line that is read in */                            |  1780
  102       |      i= 0;                   /* counter  */                                                  |  1781
  103       |  unsigned short expctid;     /*  Message ID that is expected */                              |  1782
  104       |  char commhandle??(10??),    /*  Command Line Parameter  */                                  |  1783
```

Figure 4-1 (Part 10 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                S1016897        12/19/90 08:51:20     Page   28
 Line  STMT                                                                                                          SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  105        |        *buffer,            /* Pointer to buffer     */                                       |  1784
  106        |        rmtdte??(18??),     /* Remote DTE read in       */                                    |  1785
  107        |        line??(132??),      /* Line to read in       */                                       |  1786
  108        |        key??(256??);       /* Data Queue key identifier  */                                  |  1787
  109        |   desc *descriptor;        /* Pointer to buffer descriptor  */                               |  1788
  110        |                                                                                              |  1789
  111        |   /** definitions for the API functions  **/                                                  |  1790
  112        |   enableparms enable;                                                                        |  1791
  113        |   disableparms disable;                                                                      |  1792
  114        |   sendparms send;                                                                            |  1793
  115        |   recvparms recv;                                                                            |  1794
  116        |   setfparms setf;                                                                            |  1795
  117        |   timerparms timer;                                                                          |  1796
  118        |   qlindparms qlind;                                                                          |  1797
  119        |   qentry dataq;                                                                              |  1798
  120        |   hdrparms *header;                                                                          |  1799
  121        |                                                                                              |  1800
  122        |                                                                                              |  1801
  123        |/******  Annnddddddd....  there off!!   ***********/                                          |  1802
              6
  124        |                                                                                              |  1803
  125        |   /***---  Open the file to send to remote side        ----**/                               |  1804
  126      1 |   if ((fptr = fopen("UDCS_APPLS/INFILE(INFILE))", "r")) == N                                 |  1805
  127        |      {                                                                                       |  1806
  128      2 |      printf("Unable to open source input file in UDCS_APPLS LIB.\n");                        |  1807
  129      3 |      printf("The Program was terminated.\n\n");                                              |  1808
  130      4 |      return;                                                                                 |  1809
  131        |      }                                                                                       |  1810
  132        |   /***---  Open the display file as our input screen.  ----**/                               |  1811
  133      5 |   if ((screen = fopen("ERRORSPEC", "ab+ type=record")) == NULL)                              |  1812
  134        |      {                                                                                       |  1813
  135      6 |      printf("Unable to open display file.\n");                                               |  1814
  136      7 |      printf("The Program was terminated.\n\n");                                              |  1815
  137      8 |      return;                                                                                 |  1816
  138        |      }                                                                                       |  1817
  139        |                                                                                              |  1818
  140        |   /** set the exception handler  **/                                                         |  1819
  141      9 |   signal(SIGABRT,&handler);                                                                  |  1820
  142        |                                                                                              |  1821
  143        |   /**  Clear the command line Parameters  **/                                                |  1822
  144     10 |   strncpy(enable.linename, "          ", 10);   /* Clear linename */                         |  1823
  145     11 |   strncpy(commhandle, "          ", 10);        /* Clear Commhandle */                       |  1824
  146     12 |   strncpy(rmtdte, "                 ", 17);     /* Clear Remote DTE */                       |  1825
  147        |                                                                                              |  1826
  148        |   /**  Receive command line Parameters  **/                                                  |  1827
  149     13 |   strncpy(enable.linename, argv??(1??), strlen(argv??(1??)));                                |  1828
  150     14 |   strncpy(commhandle, argv??(2??), strlen(argv??(2??)));                                     |  1829
  151     15 |   strncpy(rmtdte, argv??(3??), strlen(argv??(3??)));                                         |  1830
  152     16 |   rmtdte??(strlen(argv??(3??))??) = '\0';                                                    |  1831
  153        |                                                                                              |  1832
  154        |   /** Initialize the user spaces  **/                                                        |  1833
  155     17 |   strncpy(inbuff.library, "UDCS_APPLS", 10);    /*  Input Buffer */                          |  1834
  156     18 |   strncpy(inbuff.name, "SOURCEIBUF", 10);                                                    |  1835
  157     19 |   strncpy(indesc.library, "UDCS_APPLS", 10);    /* Input B Desc  */                          |  1836
  158     20 |   strncpy(indesc.name, "SOURCEBDSC", 10);                                                    |  1837
```

Figure 4-1 (Part 11 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                  S1016897        12/19/90 08:51:20     Page    29
 Line  STMT                                                                                                           SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  159    21 | strncpy(outbuff.library, "UDCS_APPLS", 10);   /*  Output Buffer*/                              |   1838
  160    22 | strncpy(outbuff.name, "SOURCEOBUF", 10);                                                       |   1839
  161    23 | strncpy(outdesc.library, "UDCS_APPLS", 10);   /* Output B Desc */                              |   1840
  162    24 | strncpy(outdesc.name, "SOURCEODSC", 10);                                                       |   1841
  163    25 | strncpy(qname.library, "UDCS_APPLS", 10);   /*  Data queue  */                                 |   1842
  164    26 | strncpy(qname.name, "X25DTAQ   ", 10);                                                         |   1843
  165       |                                                                                                |   1844
  166       | /*****   retrieve the line description information  ******/                                    |   1845
  167    27 | x25lind (&qlind, enable.linename);                                                             |   1846
  168       |                                                                                                |   1847
  169    28 | if ((qlind.retcode != 0) || (qlind.reason != 0))                                               |   1848
  170       |    {                                                                                           |   1849
  171    29 |    printf("Query line description failed.\n");                                                 |   1850
  172    30 |    printf("Return code = %d\n", qlind.retcode);                                                |   1851
  173    31 |    printf("Reason code = %d\n\n", qlind.reason);                                               |   1852
  174    32 |    return;                                                                                     |   1853
  175       |    }                                                                                           |   1854
  176       |                                                                                                |   1855
  177       | /*****   Hard Code the QOLELINK Input Parameters  ******/                                      |   1856
  178    33 | enable.maxdtax25 = 512;                                                                        |   1857
  179    34 | enable.keylength = 3;                                                                          |   1858
  180    35 | strncpy (enable.keyvalue, "SND", 3);                                                           |   1859
              7
  181       |                                                                                                |   1860
  182       | /**************************************************/                                           |   1861
  183       | /************  Enable the line  *******************/                                           |   1862
  184       | /**************************************************/                                           |   1863
  185       | QOLELINK (&(enable.retcode), &(enable.reason), &(enable.tdusize),\                             |   1864
  186       |     &(enable.numunits), &(enable.maxdtalan), &(enable.maxdtax25),\                             |   1865
  187       |     (char *)&inbuff, (char *)&indesc, (char *)&outbuff,\                                       |   1866
  188       |     (char *)&outdesc, &(enable.keylength), enable.keyvalue,\                                   |   1867
  189    36 |     (char *)&qname, enable.linename, commhandle);                                              |   1868
  190       |                                                                                                |   1869
  191    37 | if ((enable.retcode != 0) || (enable.reason != 0))                                             |   1870
  192       |    {                                                                                           |   1871
  193       |    printf("Line %.10s with Commhandle %.10s was NOT ENABLED.\n",\                              |   1872
  194    38 |           enable.linename, commhandle);                                                        |   1873
  195    39 |    printf("Return code = %d\n", enable.retcode);                                               |   1874
  196    40 |    printf("Reason code = %d\n\n", enable.reason);                                              |   1875
  197    41 |    return;                                                                                     |   1876
  198       |    }                                                                                           |   1877
              8
  199       |                                                                                                |   1878
  200       | /*-------- Set a timer for Enable Link ---------**/                                            |   1879
  201    42 | expctid = 0xF0F0;                                                                              |   1880
  202    43 | settimer(&expctid, "Enable", &dataq, &qname, commhandle);                                      |   1881
  203    44 | if (expctid != 0xF0F0)                                                                         |   1882
  204       |    {                                                                                           |   1883
  205    45 |    disablelink (&disable, commhandle, &qname);                                                 |   1884
  206    46 |    return;                                                                                     |   1885
  207       |    }                                                                                           |   1886
  208       |                                                                                                |   1887
              9
  209       |                                                                                                |   1888
  210       | /*****************************************************************/                            |   1889
  211       | /**************   Set up a Call Request Packet  ******************/                            |   1890
  212       | /*****************************************************************/                            |   1891
```

Figure 4-1 (Part 12 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE              S1016897        12/19/90 08:51:20      Page    30
 Line  STMT                                                                                                       SEQNBR   INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  213       |                                                                                                 |   1892
  214       |  /****   Get pointers to the user spaces.   ******/                                             |   1893
  215    47 |  QUSPTRUS(&outbuff, &buffer);                                                                   |   1894
  216    48 |  QUSPTRUS(&outdesc, &descriptor);                                                               |   1895
  217       |                                                                                                 |   1896
  218    49 |  send.ucep = 26;              /*  set the UCEP number  */                                        |   1897
  219    50 |  send.operation = 0xB000;     /*  send a call request  */                                        |   1898
  220    51 |  send.numdtaelmnts = 1;       /*  send one data unit   */                                        |   1899
  221       |                                                                                                 |   1900
  222       |  /**-----------    Send the packet   ---------**/                                                 |   1901
  223    52 |  sndformat1 (&send, buffer, rmtdte, commhandle, &qlind);                                        |   1902
  224       |                                                                                                 |   1903
  225    53 |  if ((send.retcode != 0) || (send.reason != 0))                                                 |   1904
  226       |     {                                                                                           |   1905
  227    54 |     printf("Call request packet not sent\n");                                                   |   1906
  228    55 |     printf("Return code = %d\n", send.retcode);                                                 |   1907
  229    56 |     printf("Reason code = %d\n", send.reason);                                                  |   1908
  230    57 |     printf("new pcep %d\n", send.newpcep);                                                      |   1909
  231    58 |     printespec(&(send.errorspecific));                                                          |  1910
  232       |                                                                                                 |   1911
  233    59 |     disablelink (&disable, commhandle, &qname);                                                 |   1912
  234    60 |     return;                                                                                     |   1913
  235       |     }                                                                                           |   1914
  236       |                                                                                                 |   1915
             10
  237       |                                                                                                 |   1916
  238       |  /*****************************************************************/                             |   1917
  239       |  /***********   Receive the Call CONFIRMATION packet      ********/                              |   1918
  240       |  /*****************************************************************/                             |   1919
  241       |                                                                                                 |   1920
  242       |  /*-------- Set a timer to receive a message  ---------**/                                       |   1921
  243    61 |  expctid = 0xF0F3;                                                                              |   1922
  244    62 |  settimer(&expctid, "Rcv Call", &dataq, &qname, commhandle);                                    |   1923
  245    63 |  if (expctid != 0xF0F3)                                                                         |   1924
  246       |     {                                                                                           |   1925
  247    64 |     disablelink (&disable, commhandle, &qname);                                                 |   1926
  248    65 |     return;                                                                                     |   1927
  249       |     }                                                                                           |   1928
  250       |                                                                                                 |   1929
  251       |  /** Get pointer to use space  **/                                                              |   1930
  252    66 |  QUSPTRUS (&inbuff, &buffer);                                                                   |   1931
  253    67 |  QUSPTRUS (&indesc, &descriptor);                                                               |   1932
  254       |                                                                                                 |   1933
  255       |  QOLRECV (&(recv.retcode), &(recv.reason), &(recv.ucep),\                                      |   1934
  256       |           &(recv.pcep), &(recv.operation), &(recv.numdtaunits),\                                |   1935
  257    68 |           &(recv.dataavail), &(recv.errorspecific), commhandle);                                |   1936
  258       |                                                                                                 |   1937
  259    69 |  if ((recv.retcode != 0) || (recv.reason != 0))                                                 |   1938
  260       |     {                                                                                           |   1939
  261    70 |     printf("Recv Call reqst resp failed\n");                                                    |   1940
  262    71 |     printf("return code %d\n", recv.retcode);                                                   |   1941
  263    72 |     printf("reason code %d\n", recv.reason);                                                    |   1942
  264    73 |     printespec(&(send.errorspecific));                                                          |   1943
  265       |                                                                                                 |   1944
  266    74 |     disablelink (&disable, commhandle, &qname);                                                 |   1945
```

Figure 4-1 (Part 13 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                  S1016897         12/19/90 08:51:20     Page    31
 Line  STMT                                                                                                            SEQNBR   INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  267    75 |    return;                                                                                     |  1946
  268       |    }                                                                                           |  1947
  269       |                                                                                                |  1948
  270       |  /* Interpret the Received Operation  */                                                       |  1949
  271    76 |  if (recv.operation != 0xB001)                                                                 |  1950
  272       |    {                                                                                           |  1951
  273    77 |    printf("Recvd opr %x instead of opr B001\n", recv.operation);                               |  1952
  274    78 |    disablelink (&disable, commhandle, &qname);                                                 |  1953
  275    79 |    return;                                                                                     |  1954
  276       |    }                                                                                           |  1955
  277       |                                                                                                |  1956
  278    80 |  printf("We have an X.25 SVC connection\n\n");                                                 |  1957
  279       |                                                                                                |  1958
             11
  280       |                                                                                                |  1959
  281       |  /*****************************************************************/                            |  1960
  282       |  /***************   Send the file to the target application *******/                            |  1961
  283       |  /*****************************************************************/                            |  1962
  284       |                                                                                                |  1963
  285    81 |  send.pcep = send.newpcep;       /*  set the PCEP number  */                                   |  1964
  286       |                                                                                                |  1965
  287       |  /***************   Send the Mbr LGRF in file DOC  *****************/                           |  1966
  288    82 |  linesiz = getline(line, 92, fptr);   /* Get first record  **/                                 |  1967
  289    83 |  while (linesiz != 0)                                                                          |  1968
  290       |    {                                                                                           |  1969
  291       |    /***************   Send a Packet of Data    ***************/                                 |  1970
  292       |                                                                                                |  1971
  293       |    /****   Get pointers to the user spaces.   ******/                                          |  1972
  294    84 |    QUSPTRUS(&outbuff, &buffer);                                                                |  1973
  295    85 |    QUSPTRUS(&outdesc, &descriptor);                                                            |  1974
  296       |                                                                                                |  1975
  297    86 |    send.operation = 0x0000;                                                                    |  1976
  298    87 |    send.numdtaelmnts = 1;                                                                      |  1977
  299       |                                                                                                |  1978
  300       |    /**-----   Send the packet      -------------**/                                            |  1979
  301    88 |    senddata (&send, buffer, descriptor, commhandle, line, linesiz);                            |  1980
  302       |                                                                                                |  1981
  303    89 |    if ((send.retcode != 0) || (send.reason != 0))                                              |  1982
  304       |      {                                                                                         |  1983
  305    90 |      printf("Data NOT sent for commhandle %.9s\n", commhandle);                                |  1984
  306    91 |      printf("Return code = %d\n", send.retcode);                                               |  1985
  307    92 |      printf("Reason code = %d\n", send.reason);                                                |  1986
  308    93 |      printf("new pcep %d\n", send.newpcep);                                                    |  1987
  309    94 |      printespec(&(send.errorspecific));                                                        |  1988
  310       |                                                                                                |  1989
  311    95 |      disablelink (&disable, commhandle, &qname);                                               |  1990
  312    96 |      return;                                                                                   |  1991
  313       |      }                                                                                         |  1992
  314    97 |    i = i + 1;                                                                                  |  1993
  315    98 |    printf("Data %d Sent for commhandle %.9s.\n\n", i, commhandle);                             |  1994
  316       |                                                                                                |  1995
  317    99 |    linesiz = getline(line, 92, fptr);  /** Get next record  **/                                |  1996
  318       |    }                      /***  End While loop  ***/                                           |  1997
  319       |                                                                                                |  1998
  320       |                                                                                                |  1999
```

Figure 4-1 (Part 14 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400         UDCS_APPLS/SOURCE              S1016897        12/19/90 08:51:20     Page    32
 Line  STMT                                                                                                    SEQNBR   INCNO
              *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  321       | /*****************************************************************/                        |  2000
  322       | /***************   Set up a Clear Request Packet  *****************/                        |  2001
  323       | /*****************************************************************/                        |  2002
             12
  324       |                                                                                             |  2003
  325       | /****   Get pointers to the user spaces.   ******/                                          |  2004
  326   100 | QUSPTRUS(&outbuff, &buffer);                                                                |  2005
  327   101 | QUSPTRUS(&outdesc, &descriptor);                                                            |  2006
  328       |                                                                                             |  2007
  329   102 | send.operation = 0xB100;         /**  send clear request  **/                               |  2008
  330   103 | send.numdtaelmnts = 1;           /**  send one data unit  **/                               |  2009
  331       |                                                                                             |  2010
  332       | /**-----------    Send the packet   ---------**/                                            |  2011
  333   104 | sndformat2 (&send, buffer, commhandle);                                                     |  2012
  334       |                                                                                             |  2013
  335   105 | if ((send.retcode != 0) || (send.reason != 0))                                              |  2014
  336       |    {                                                                                        |  2015
  337   106 |    printf("Clear request packet not sent\n");                                               |  2016
  338   107 |    printf("Return code = %d\n", send.retcode);                                              |  2017
  339   108 |    printf("Reason code = %d\n", send.reason);                                               |  2018
  340   109 |    printf("new pcep %d\n", send.newpcep);                                                   |  2019
  341   110 |    printespec(&(send.errorspecific));                                                       |  2020
  342       |                                                                                             |  2021
  343   111 |    disablelink (&disable, commhandle, &qname);                                              |  2022
  344   112 |    return;                                                                                  |  2023
  345       |    }                                                                                        |  2024
  346       |                                                                                             |  2025
             13
  347       |                                                                                             |  2026
  348       | /*****************************************************************/                        |  2027
  349       | /***********   Receive the Clear Request Response packet   ********/                        |  2028
  350       | /*****************************************************************/                        |  2029
  351       |                                                                                             |  2030
  352       | /*-------- Set a timer to receive a message  ---------**/                                   |  2031
  353   113 | expctid = 0xF0F3;                                                                           |  2032
  354   114 | settimer(&expctid, "Rv Clr Rqt", &dataq, &qname, commhandle);                               |  2033
  355   115 | if (expctid != 0xF0F3)                                                                      |  2034
  356       |    {                                                                                        |  2035
  357   116 |    disablelink (&disable, commhandle, &qname);                                              |  2036
  358   117 |    return;                                                                                  |  2037
  359       |    }                                                                                        |  2038
  360       |                                                                                             |  2039
  361       |                                                                                             |  2040
  362       | /***********   Call QOLRECV to Receive the Clear Response   ********/                        |  2041
  363       | /****   Get pointers to the user spaces.   ******/                                          |  2042
  364   118 | QUSPTRUS (&inbuff, &buffer);                                                                |  2043
  365   119 | QUSPTRUS (&indesc, &descriptor);                                                            |  2044
  366       |                                                                                             |  2045
  367       | QOLRECV (&(recv.retcode), &(recv.reason), &(recv.ucep),\                                    |  2046
  368       |         &(recv.pcep), &(recv.operation), &(recv.numdtaunits),\                              |  2047
  369   120 |         &(recv.dataavail), &(recv.errorspecific), commhandle);                              |  2048
  370       |                                                                                             |  2049
  371   121 | if ((recv.retcode != 0) || (recv.reason != 0))                                              |  2050
  372       |    {                                                                                        |  2051
  373   122 |    printf("Recv clear response failed\n");                                                  |  2052
  374   123 |    printf("return code %d\n", recv.retcode);                                                |  2053
```

Figure 4-1 (Part 15 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                    S1016897        12/19/90 08:51:20    Page    33
 Line  STMT                                                                                                            SEQNBR  INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  375   124 |      printf("reason code %d\n", recv.reason);                                                      |   2054
  376   125 |      printespec(&(send.errorspecific));                                                            |   2055
  377       |                                                                                                    |   2056
  378   126 |      disablelink (&disable, commhandle, &qname);                                                   |   2057
  379   127 |      return;                                                                                       |   2058
  380       |      }                                                                                             |   2059
  381       |                                                                                                    |   2060
  382       |   /* Interpret the Received Operation  */                                                          |   2061
  383   128 |   if (recv.operation != 0xB101)                                                                    |   2062
  384       |      {                                                                                             |   2063
  385   129 |      printf("Recvd opr %x instead of opr B101\n", recv.operation);                                 |   2064
  386   130 |      disablelink (&disable, commhandle, &qname);                                                   |   2065
  387   131 |      return;                                                                                       |   2066
  388       |      }                                                                                             |   2067
  389       |                                                                                                    |   2068
  390       |   /***********************************************/                                              |   2069
  391       |   /***  Disable the link and end the program  ****/                                              |   2070
  392       |   /***********************************************/                                              |   2071
  393   132 |   disablelink (&disable, commhandle, &qname);                                                      |   2072
  394       |                                                                                                    |   2073
  395   133 |   printf("******   SSNDTH completed successfully   ******\n\n");                                    |   2074
  396       |                                                                                                    |   2075
  397       |}     /* End Main */                                                                                |   2076
  398       |                                                                                                    |   2077
  399       |                                                                                                    |   2078
  400       |/*****************************************************************/                                |   2079
  401       |/************** Start Subroutine Section    **********************/                                |   2080
  402       |/*****************************************************************/                                |   2081
  403       |                                                                                                    |   2082
  404       |/**************************************************************/                                   |   2083
  405       |/***************      Send a Packet of Data  ******************/                                    |   2084
             14
  406       |                                                                                                    |   2085
  407       |void senddata (sendparms *send,                                                                     |   2086
  408       |              char *buffer,                                                                         |   2087
  409       |              desc *descriptor,                                                                     |   2088
  410       |              char *commhandle,                                                                     |   2089
  411       |              char *line,                                                                           |   2090
  412       |              int linesiz)                                                                          |   2091
  413       |                                                                                                    |   2092
  414       |{                                                                                                   |   2093
  415     1 |  descriptor->length = linesiz;                                                                     |   2094
  416     2 |  descriptor->more = 0;                                                                             |   2095
  417     3 |  descriptor->qualified = 0;                                                                        |   2096
  418     4 |  descriptor->interrupt = 0;                                                                        |   2097
  419     5 |  descriptor->dbit = 0;                                                                             |   2098
  420     6 |  strncpy (buffer, line, linesiz);                                                                  |   2099
  421       |                                                                                                    |   2100
  422       |  QOLSEND (&(send->retcode), &(send->reason), &(send->errorspecific),\                              |   2101
  423       |           &(send->newpcep), &(send->ucep), &(send->pcep), \                                       |   2102
  424     7 |           commhandle, &(send->operation), &(send->numdtaelmnts));                                 |   2103
  425       |                                                                                                    |   2104
  426       |}  /* End senddata Subroutine  */                                                                   |   2105
  427       |                                                                                                    |   2106
  428       |                                                                                                    |   2107
```

Figure 4-1 (Part 16 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400           UDCS_APPLS/SOURCE                  S1016897        12/19/90 08:51:20       Page    34
 Line  STMT                                                                                                             SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  429       |/*******************************************************/                                          |  2108
  430       |/************  Routine to fill X.25 Format I   ***********/                                       |  2109
  431       |                                                                                                   |  2110
  432       |void sndformat1 (sendparms *send,                                                                  |  2111
  433       |                 char *buffer,                                                                     |  2112
  434       |                 char *rmtdte,                                                                     |  2113
  435       |                 char *commhandle,                                                                 |  2114
  436       |                 qlindparms *qlind)                                                                |  2115
  437       |                                                                                                   |  2116
  438       |{                                                                                                  |  2117
  439       |  format1 *output = (format1 *) buffer;                                                            |  2118
  440       |  register int counter;                                                                            |  2119
  441       |  register querydata *qd;                                                                          |  2120
  442       |                                                                                                   |  2121
  443     1 |  qd = (querydata *)&(qlind->userbuffer);                                                          |  2122
  444     2 |  output->type = 2;                /*  SVC used  */                                               |  2123
  445     3 |  output->logchanid = 0x0;                                                                         |  2124
  446     4 |  output->sendpacksize = qd->x25data.defsend;                                                      |  2125
  447     5 |  output->sendwindsize = qd->x25data.windowsend;                                                   |  2126
  448     6 |  output->recvpacksize = qd->x25data.defrecv;                                                      |  2127
  449     7 |  output->recvwindsize = qd->x25data.windowrecv;                                                   |  2128
  450       |                                                                                                   |  2129
  451     8 |  output->dtelength = strlen(rmtdte);                                                              |  2130
  452     9 |  byte(output->dte, 16, rmtdte, strlen(rmtdte));                                                   |  2131
  453    10 |  output->dbit = 0;                                                                                |  2132
  454    11 |  output->cug = 0;                                                                                 |  2133
  455    12 |  output->cugid = 0;                                                                               |  2134
  456    13 |  output->reverse = 0;                                                                             |  2135
  457    14 |  output->fast = 0;                                                                                |  2136
  458    15 |  output->faclength = 0;                                                                           |  2137
  459    16 |  byte(output->facilities, 109, "", 0);                                                            |  2138
  460    17 |  output->calllength = 1;                                                                          |  2139
  461    18 |  byte(output->callud, 128, "21", 2);   /* Contains Remote PID */                                  |  2140
  462    19 |  output->misc??(0??) = 0;     /* change to 0x80 for reset support */                              |  2141
  463    20 |  output->misc??(1??) = 0;                                                                         |  2142
  464    21 |  output->misc??(2??) = 0;                                                                         |  2143
  465    22 |  output->misc??(3??) = 0;                                                                         |  2144
  466    23 |  output->maxasmsize = 16383;                                                                      |  2145
  467    24 |  output->autoflow = 32;                                                                           |  2146
  468       |                                                                                                   |  2147
  469       |                                                                                                   |  2148
  470       |  QOLSEND (&(send->retcode), &(send->reason), &(send->errorspecific),\                             |  2149
  471       |           &(send->newpcep), &(send->ucep), &(send->pcep),\                                        |  2150
  472    25 |           commhandle, &(send->operation), &(send->numdtaelmnts));                                 |  2151
  473       |                                                                                                   |  2152
  474       |}  /* End sndformat1 Subroutine  */                                                                |  2153
  475       |                                                                                                   |  2154
  476       |                                                                                                   |  2155
  477       |/*******************************************************/                                          |  2156
  478       |/************  Routine to fill X.25 Format II   ***********/                                      |  2157
  479       |                                                                                                   |  2158
  480       |void sndformat2 (sendparms *send,                                                                  |  2159
  481       |                 char *buffer,                                                                     |  2160
  482       |                 char *commhandle)                                                                 |  2161
```

Figure 4-1 (Part 17 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                   S1016897        12/19/90 08:51:20     Page    35
 Line  STMT                                                                                                           SEQNBR  INCNO
           *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  483      |                                                                                            |   2162
  484      |{                                                                                           |   2163
  485      |  format2 *output = (format2 *) buffer;                                                     |   2164
  486      |                                                                                            |   2165
  487    1 |  output->type = 1;                                                                         |   2166
  488    2 |  output->cause = 'FF';                                                                     |   2167
  489    3 |  output->diagnostic = 'FF';                                                                |   2168
  490    4 |  output->faclength = 0;                                                                    |   2169
  491    5 |  byte(output->facilities, 109, "", 0);                                                     |   2170
  492    6 |  output->length = 0;                                                                       |   2171
  493    7 |  byte(output->userdata, 128, "", 0);                                                       |   2172
  494      |                                                                                            |   2173
  495      |                                                                                            |   2174
  496      |  QOLSEND (&(send->retcode), &(send->reason), &(send->errorspecific),\                      |   2175
  497      |          &(send->newpcep), &(send->ucep), &(send->pcep),\                                  |   2176
  498    8 |          commhandle, &(send->operation), &(send->numdtaelmnts));                           |   2177
  499      |                                                                                            |   2178
  500      |}  /* End sndformat2 Subroutine  */                                                         |   2179
  501      |                                                                                            |   2180
           15
  502      |                                                                                            |   2181
  503      |/********************************************/                                              |   2182
  504      |/********    Routine to disable    ***********/                                              |   2183
  505      |                                                                                            |   2184
  506      |void disablelink (disableparms *disable,                                                    |   2185
  507      |                  char *commhandle,                                                         |   2186
  508      |                  usrspace *qname)                                                          |   2187
  509      |                                                                                            |   2188
  510      |{                                                                                           |   2189
  511      |unsigned short expctid;                                                                     |   2190
  512      |qentry dataq;                                                                               |   2191
  513      |                                                                                            |   2192
  514    1 |  disable->vary = 1;     /*  Hard coded to be varied off  */                                |   2193
  515      |                                                                                            |   2194
  516      |  QOLDLINK (&(disable->retcode), &(disable->reason),\                                       |   2195
  517    2 |                        commhandle, &(disable->vary));                                      |   2196
  518      |                                                                                            |   2197
  519    3 |  if ((disable->retcode != 0) && (disable->reason != 00))                                   |   2198
  520      |     {                                                                                      |   2199
  521    4 |     printf ("Link %.10s did not disabled.\n", commhandle);                                 |   2200
  522    5 |     printf ("return code = %d\n", disable->retcode);                                       |   2201
  523    6 |     printf ("reason code = %d\n\n", disable->reason);                                      |   2202
  524      |     }                                                                                      |   2203
  525      |                                                                                            |   2204
  526      |  /**-------  Set a timer to receive disable complete msg --------**/                        |   2205
  527    7 |  expctid = 0xF0F1;                                                                         |   2206
  528    8 |  settimer(&expctid, "Disable", &dataq,  qname, commhandle);                                |   2207
  529    9 |  if (expctid != 0xF0F1)                                                                    |   2208
  530      |     {                                                                                      |   2209
  531   10 |     printf("Disable link did not complete successfully");                                  |   2210
  532   11 |     return;                                                                                |   2211
  533      |     }                                                                                      |   2212
  534      |                                                                                            |   2213
  535   12 |  printf ("%.10s link disabled \n", commhandle);                                            |   2214
  536      |                                                                                            |   2215
```

Figure 4-1 (Part 18 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400           UDCS_APPLS/SOURCE               S1016897        12/19/90 08:51:20     Page    36
 Line  STMT                                                                                                          SEQNBR   INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  537        |  /**  close the files  **/                                                                    |    2216
  538     13 |  fclose(fptr);                                                                                |    2217
  539     14 |  fclose(screen);                                                                              |    2218
  540        |                                                                                               |    2219
  541        |}  /* End disablelink Subroutine  */                                                           |    2220
  542        |                                                                                               |    2221
  543        |                                                                                               |    2222
  544        |/**************************************************************/                              |    2223
  545        |/**    Routine to convert string to Hexidecimal format   ******/                                 |    2224
  546        |                                                                                               |    2225
  547        |void byte (char *dest,                                                                         |    2226
  548        |           int dlength,                                                                        |    2227
  549        |           char *source,                                                                       |    2228
  550        |           int slength)                                                                        |    2229
  551        |                                                                                               |    2230
  552        |{                                                                                              |    2231
  553        |  register int counter;                                                                        |    2232
  554        |  char holder??(2??);                                                                          |    2233
  555        |                                                                                               |    2234
  556      1 |  for (counter=0;counter<dlength;counter++)                                                    |    2235
  557      2 |   dest??(counter??)=0;                                                                        |    2236
  558      3 |  for (counter=slength-1;counter>=0;counter--)                                                 |    2237
  559      4 |   if isxdigit(source??(counter??))                                                            |    2238
  560        |     {                                                                                         |    2239
  561      5 |      holder??(0??)=source??(counter??);                                                       |    2240
  562      6 |      holder??(1??)='\0';                                                                      |    2241
  563      7 |      if  (counter % 2 == 0)                                                                   |    2242
  564      8 |        dest??(counter/2??) += (char) hextoint(holder)*16;                                     |    2243
  565      9 |      else dest??(counter/2??) += (char) hextoint(holder);                                     |    2244
  566        |     }                                                                                         |    2245
  567        |                                                                                               |    2246
  568        |}  /* End byte Subroutine  */                                                                  |    2247
  569        |                                                                                               |    2248
  570        |                                                                                               |    2249
  571        |/**************************************************************/                              |    2250
  572        |/**    Routine to display the ErrorSpecific output       ******/                                 |    2251
  573        |                                                                                               |    2252
  574        |void printespec(espec *errorspecific)                                                          |    2253
  575        |                                                                                               |    2254
  576        |{                                                                                              |    2255
  577        |  especout outparms;                                                                           |    2256
  578        |                                                                                               |    2257
  579      1 |  QXXFORMAT(screen, "ERRORSPEC ");                                                             |    2258
  580      2 |  sprintf(outparms.hwecode, "%.8X", errorspecific->hwecode);                                   |    2259
  581        |  sprintf(outparms.timestamp, "%.8X%.8X", errorspecific->timestamphi,\                         |    2260
  582      3 |                                        errorspecific->timestamplo);                           |    2261
  583      4 |  sprintf(outparms.elogid, "%.8X", errorspecific->elogid);                                     |    2262
  584      5 |  if (errorspecific->flags & 0x40)                                                             |    2263
  585      6 |    outparms.fail = 'Y';                                                                       |    2264
  586      7 |  else outparms.fail = 'N';                                                                    |    2265
  587      8 |  if (errorspecific->flags & 0x20)                                                             |    2266
  588      9 |    outparms.zerocodes = 'Y';                                                                  |    2267
  589     10 |  else outparms.zerocodes = 'N';                                                               |    2268
  590     11 |  if (errorspecific->flags & 0x10)                                                             |    2269
```

Figure 4-1 (Part 19 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                 S1016897         12/19/90 08:51:20     Page    37
 Line  STMT                                                                                                          SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  591    12 |    outparms.qsysopr = 'Y';                                                                     |   2270
  592    13 |  else outparms.qsysopr = 'N';                                                                  |   2271
  593    14 |  sprintf(outparms.cause,"%.2X", errorspecific->cause);                                         |   2272
  594    15 |  sprintf(outparms.diagnostic, "%.2X", errorspecific->diagnostic);                              |   2273
  595    16 |  sprintf(outparms.erroffset, "%.6d", errorspecific->erroroffset);                              |   2274
  596    17 |  fwrite(&outparms, 1, sizeof(especout), screen);                                               |   2275
  597    18 |  fread("", 0, 0, screen);                                                                      |   2276
  598       |                                                                                               |   2277
  599       |}  /* End printespec Subroutine  */                                                            |   2278
  600       |                                                                                               |   2279
  601       |/*************   Set a timer and dequeue next entry  ******/                                    |   2280
            16
  602       |                                                                                               |   2281
  603       |void settimer (unsigned short *expctid,                                                        |   2282
  604       |               char *process,                                                                  |   2283
  605       |               qentry *dataq,                                                                  |   2284
  606       |               usrspace *qname,                                                                |   2285
  607       |               char *commhandle)                                                               |   2286
  608       |                                                                                               |   2287
  609       |{                                                                                              |   2288
  610       |timerparms timer;                                                                              |   2289
  611       |disableparms disable;                                                                          |   2290
  612       |int length;                                                                                    |   2291
  613       |char key??(6??);                                                                               |   2292
  614       |                                                                                               |   2293
  615     1 |  timer.interval = 20000;       /*  Set timer for 20 seconds */                                |   2294
  616     2 |  timer.establishcount = 1;                                                                    |   2295
  617     3 |  timer.keylength = 6;          /*  No key value  */                                           |   2296
  618     4 |  strncpy(timer.keyvalue, "SOURCE", 6);                                                        |   2297
  619     5 |  timer.operation = 1;           /*  Set a timer  */                                           |   2298
  620       |                                                                                               |   2299
  621       |  QOLTIMER (&(timer.retcode), &(timer.reason), timer.handleout,\                               |   2300
  622       |           timer.handlein, (char *)qname, &(timer.operation),\                                 |   2301
  623       |           &(timer.interval), &(timer.establishcount),\                                        |   2302
  624     6 |           &(timer.keylength), timer.keyvalue, timer.userdata);                                |   2303
  625       |                                                                                               |   2304
  626     7 |  if ((timer.retcode != 0) || (timer.reason != 0))                                             |   2305
  627       |     {                                                                                         |   2306
  628     8 |     printf("%s timer failed while being set.\n", process);                                    |   2307
  629     9 |     printf("Return code = %d\n", timer.retcode);                                              |   2308
  630    10 |     printf("Reason code = %d\n\n", timer.reason);                                             |   2309
  631       |     }                                                                                         |   2310
  632       |                                                                                               |   2311
  633       |/**-------  Dequeue an entry  --------**/                                                      |   2312
  634    11 |  strncpy(key, "SOURCE",6);                                                                    |   2313
  635    12 |  length = 6;                                                                                  |   2314
  636    13 |  dequeue (length, key, dataq, qname);                                                         |   2315
  637       |                                                                                               |   2316
  638       |  /*** Cancel timer  ***/                                                                      |   2317
  639    14 |  if (dataq->msgid != 0xF0F4)                                                                  |   2318
  640       |     {                                                                                         |   2319
  641    15 |     strncpy(timer.handlein, timer.handleout, 8);                                              |   2320
  642    16 |     timer.operation = 2;           /*  Set one timer  */                                      |   2321
  643       |                                                                                               |   2322
  644       |     QOLTIMER (&(timer.retcode), &(timer.reason), timer.handleout,\                            |   2323
```

Figure 4-1 (Part 20 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE            S1016897        12/19/90 08:51:20     Page    38
 Line  STMT                                                                                                      SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  645        |          timer.handlein, (char *)qname, &(timer.operation),\                              |   2324
  646        |          &(timer.interval), &(timer.establishcount),\                                     |   2325
  647    17  |          &(timer.keylength), timer.keyvalue, timer.userdata);                             |   2326
  648        |                                                                                           |   2327
  649    18  |     if ((timer.retcode != 0) || (timer.reason != 0))                                      |   2328
  650        |        {                                                                                  |   2329
  651    19  |        printf("%s timer failed while being canceled\n", process);                         |   2330
  652    20  |        printf("Return code = %d\n", timer.retcode);                                       |   2331
  653    21  |        printf("Reason code = %d\n\n", timer.reason);                                      |   2332
  654        |        }                                                                                  |   2333
  655        |     }                                                                                     |   2334
  656        |                                                                                           |   2335
  657    22  |  if (dataq->msgid != *expctid)                                                            |   2336
  658        |      {                                                                                    |   2337
  659        |      printf ("A %.4X message ID was received instead of %.4X\n",\                         |   2338
  660    23  |             dataq->msgid, *expctid);                                                      |   2339
  661    24  |      printf ("%s completion message was not received\n", process);                        |   2340
  662    25  |      *expctid = dataq->msgid;                                                             |   2341
  663        |      }                                                                                    |   2342
  664        |                                                                                           |   2343
  665        |}  /* End settimer Subroutine  */                                                          |   2344
  666        |                                                                                           |   2345
  667        |                                                                                           |   2346
  668        |/***************************************************************/                         |   2347
  669        |/*******   Dequeues the Incoming Message and processes it ******/                          |   2348
  670        |                                                                                           |   2349
  671        |void dequeue (int length,                                                                  |   2350
  672        |             char *key,                                                                    |   2351
  673        |             qentry *dataq,                                                                |   2352
  674        |             usrspace *qname)                                                              |   2353
  675        |                                                                                           |   2354
  676        |{                                                                                          |   2355
  677        |  char fldlen??(3??),                                                                      |   2356
  678        |       waittime??(3??),                                                                    |   2357
  679        |       keylen??(2??),                                                                      |   2358
  680        |       senderid??(2??),                                                                    |   2359
  681        |       *pointer,                                                                           |   2360
  682        |       order??(2??);                                                                       |   2361
  683        |  register int counter;                                                                    |   2362
  684        |                                                                                           |   2363
  685        |                                                                                           |   2364
  686     1  |  waittime??(0??) = 0;                                                                     |   2365
  687     2  |  waittime??(1??) = 0;                                                                     |   2366
  688     3  |  waittime??(2??) = 0x1D;   /*  Hard code a delay of infinite   */                        |   2367
  689     4  |  keylen??(0??) = 0;                                                                       |   2368
  690     5  |  keylen??(1??) = 0x6F;   /* Hard code a keylength of 6  */                                |   2369
  691     6  |  senderid??(0??) = 0;                                                                     |   2370
  692     7  |  senderid??(1??) = 0x0F;                                                                  |   2371
  693     8  |  strncpy(order, "EQ", 2);                                                                 |   2372
  694        |                                                                                           |   2373
  695     9  |  fflush(stdin);                                                                           |   2374
  696    10  |  pointer = (char *)dataq;                                                                 |   2375
  697    11  |  for (counter = 0; counter < 336; counter++)                                              |   2376
  698    12  |    pointer??(counter??) = 0;                                                              |   2377
```

Figure 4-1 (Part 21 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                  S1016897        12/19/90 08:51:20     Page   39
 Line  STMT                                                                                                         SEQNBR  INCNO
           *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  699          |                                                                                            |   2378
  700     13   |  strncpy (dataq->type, "       ", 7);                                                      |   2379
  701     14   |  while ((strncmp(dataq->type, "*USRDFN", 7) != 0) || (fldlen == 0))                        |   2380
  702          |     QRCVDTAQ(qname->name, qname->library, fldlen, dataq, waittime,\                        |   2381
  703     15   |           order, keylen, key, senderid,"");                                                |   2382
  704          |                                                                                            |   2383
  705          |}  /* End dequeue Subroutine  */                                                            |   2384
  706          |                                                                                            |   2385
  707          |                                                                                            |   2386
  708          |/************************************************************/                              |   2387
  709          |/**  x25lind:  Read a record into buf and return length   **/                                 |   2388
               17
  710          |                                                                                            |   2389
  711          |void x25lind (qlindparms *qlind, char *linename)                                            |   2390
  712          |{                                                                                           |   2391
  713          |register int counter;                                                                       |   2392
  714          |                                                                                            |   2393
  715      1   |  for(counter=0;counter<256;counter++)                                                      |   2394
  716      2   |     qlind->userbuffer??(counter??)=0;                                                      |   2395
  717          |                                                                                            |   2396
  718      3   |  qlind->format = 0x01;                                                                     |   2397
  719          |  QOLQLIND (&(qlind->retcode), &(qlind->reason), &(qlind->nbytes),\                         |   2398
  720      4   |            qlind->userbuffer, linename, &(qlind->format));                                 |   2399
  721          |                                                                                            |   2400
  722          |}  /*  End x25lind Subroutine  */                                                           |   2401
  723          |                                                                                            |   2402
  724          |                                                                                            |   2403
  725          |/*******************************************************/                                   |   2404
  726          |/**  Getline:  Read a record into line and return length   **/                              |   2405
  727          |                                                                                            |   2406
  728          |int getline (char *line, int max, FILE *fptr)                                               |   2407
  729          |{                                                                                           |   2408
  730      1   |   if (fgets(line, max, fptr) == NULL)                                                      |   2409
  731      2   |      return 0;                                                                             |   2410
  732          |   else                                                                                     |   2411
  733      3   |      return strlen(line);                                                                  |   2412
  734          |                                                                                            |   2413
  735          |}  /* End getline Subroutine  */                                                            |   2414
  736          |                                                                                            |   2415
  737          |/***************************************************/                                       |   2416
  738          |/*  Exception handler, so that a failure will not                                            |   2417
  739          |    kill the program, and any associated data!!  */                                         |   2418
               18
  740          |                                                                                            |   2419
  741          |void handler (disableparms disable, usrspace *qname)                                        |   2420
  742          |{                                                                                           |   2421
  743          |  sigdata_t *data;                                                                          |   2422
  744          |                                                                                            |   2423
  745      1   |  disablelink(&disable, "*ALL      ", qname);                                               |   2424
  746      2   |  printf("The program received an excecption.\n");                                         |   2425
  747      3   |  printf("Disable Link was called & the program was terminated.\n\n");                      |   2426
  748          |                                                                                            |   2427
  749      4   |  data=sigdata();                                                                           |   2428
  750      5   |  data->sigact->xhalt=0;                                                                    |   2429
  751      6   |  data->sigact->xrtntosgnler=0;                                                             |   2430
  752      7   |  data->sigact->xresigprior=0;                                                              |   2431
```

Figure 4-1 (Part 22 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                  S1016897        12/19/90 08:51:20     Page   40
 Line  STMT                                                                                                         SEQNBR  INCNO
           *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  753      8   |  data->sigact->xresigouter=0;                                                              |   2432
  754          |                                                                                            |   2433
  755          |}  /*  End handler Subroutine  */                                                           |   2434
                                     * * * * *   E N D   O F   S O U R C E   * * * * *
```

Figure 4-1 (Part 23 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                   S1016897         12/19/90 08:51:20     Page    41
                                             * * * * *   I N C L U D E S   * * * * *
INCNO   Include Name                    Last change         Actual Include Name
    1   header                          90/12/19 08:49:31   UDCS_APPLS/QCSRC/HEADER
    2   typedefs                        90/12/19 08:49:31   UDCS_APPLS/QCSRC/TYPEDEFS
    3   stdio.h                         90/11/12 17:24:55   QCC/H/STDIO
    4   stddef.h                        90/11/12 17:24:54   QCC/H/STDDEF
    5   errno.h                         90/11/12 17:24:49   QCC/H/ERRNO
    6   signal.h                        90/11/12 17:24:53   QCC/H/SIGNAL
    7   ctype.h                         90/11/12 17:24:49   QCC/H/CTYPE
    8   stdarg.h                        90/11/12 17:24:54   QCC/H/STDARG
    9   stdlib.h                        90/11/12 17:24:56   QCC/H/STDLIB
   10   signal.h                        90/11/12 17:24:53   QCC/H/SIGNAL
   11   xxasio.h                        90/11/12 17:24:57   QCC/H/XXASIO
   12   xxcvt.h                         90/11/12 17:24:58   QCC/H/XXCVT
   13   string.h                        90/11/12 17:24:56   QCC/H/STRING
   14   ctype.h                         90/11/12 17:24:49   QCC/H/CTYPE
   15   hexconv                         90/12/19 08:49:27   UDCS_APPLS/QCSRC/HEXCONV
   16   stdio.h                         90/11/12 17:24:55   QCC/H/STDIO
                                      * * * * *   E N D   O F   I N C L U D E S   * * * * *
```

Figure 4-1 (Part 24 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                   S1016897         12/19/90 08:51:20     Page    42
                                        * * * * *   M E S S A G E   S U M M A R Y   * * * * *
     Total       Info(0-4)           Warning(5-19)        Error(20-29)         Severe(30-39)        Terminal(40-99)
         0           0                     0                   0                    0                    0
                                   * * * * *   E N D   O F   M E S S A G E   S U M M A R Y   * * * * *
```

Figure 4-1 (Part 25 of 26). C/400 Compiler Listing for the Source Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/SOURCE                   S1016897         12/19/90 08:51:20     Page  43
 ROUTINE          BLOCK NUMBER  SCOPE  TYPE
 <MAIN>                  2      LOCAL  MAIN-PROGRAM
 __sigdata               6      LOCAL  PROCEDURE
 __sgnl                 12      LOCAL  PROCEDURE
 __frdinit              49      LOCAL  PROCEDURE
 __getrec               50      LOCAL  PROCEDURE
 __putrec               51      LOCAL  PROCEDURE
 __frdexit              52      LOCAL  PROCEDURE
 __fwrinit              53      LOCAL  PROCEDURE
 __fwrexit              54      LOCAL  PROCEDURE
QXXFORMAT              129      LOCAL  PROCEDURE
 __strlen              164      LOCAL  PROCEDURE
 __strncmp             168      LOCAL  PROCEDURE
 __strncpy             170      LOCAL  PROCEDURE
 inttohex              181      ENTRY   PROCEDURE
 hextoint              182      ENTRY   PROCEDURE
 senddata              196      ENTRY   PROCEDURE
 sndformat1            197      ENTRY   PROCEDURE
 sndformat2            198      ENTRY   PROCEDURE
 byte                  200      ENTRY   PROCEDURE
 printespec            201      ENTRY   PROCEDURE
 settimer              202      ENTRY   PROCEDURE
 dequeue               203      ENTRY   PROCEDURE
 x25lind               204      ENTRY   PROCEDURE
 getline               205      ENTRY   PROCEDURE
 disablelink           206      ENTRY   PROCEDURE
 handler               207      ENTRY   PROCEDURE
 main                  208      ENTRY   PROCEDURE
Program SOURCE was created in library UDCS_APPLS.
                                  * * * * *   E N D   O F   C O M P I L A T I O N   * * * * *
```

Figure 4-1 (Part 26 of 26). C/400 Compiler Listing for the Source Application

The block numbers and explanations below correspond to those in the source application's program listing.

1 Some general C structure declarations used by both the source and target application programs.

Note: The example programs of this chapter were developed using a 5250 display station and keyboard where the left ([) and right (]) bracket characters are not supported. As a result of this, C language array declarations used in the applications use the character sequence of "??(" to denote a left bracket and "??)" to denote a right bracket.

You do not need to change C language applications containing array declarations with the bracket characters in order to compile and run on the AS/400.

2 C/400 compiler directives are used here to indicate that standard OS/400 linkage conventions should be used when calling the user-defined communications support APIs.

3 C external function definitions for the user-defined communications support APIs. Note that all parameters are passed by reference.

4 More C structure declarations that are used when calling the user-defined communications support APIs.

5 Function prototypes of the internal functions used in this program.

6 Call the C library routines fopen() and signal() to open the source file and set up a signal handler to process AS/400 exceptions, respectively. An example of an exception would be accessing a data area with a NULL pointer. If an exception situation is encountered, the handler() will be called in order for the program to end.

7 Call the QOLQLIND API to retrieve local configuration information from the AS/400 line description about that will be used for communications. Next, call the QOLELINK API to enable the line description using the line name and communications handle passed as input parameters to this program.

8 Call the QOLTIMER API to time the completion of the enable link operation. If the timer expires before the enable-complete entry is posted on the this program's data queue, then this program will end.

9 Call the QOLSEND API with a X'B000' operation to establish a connection to the target application program.

10 Monitor the source program's data queue for the call confirmation. The source program will be notified of the call confirmation by call the QOLRECV API and receiving a X'B001' operation in the program's input buffer.

11 This is the main send loop for the source program. The data from the source file is placed one line at a time in the output buffer and then the QOLSEND API is called to send one data unit of the file to System B. This process repeats until the contents of the entire file have been transmitted to the target application.

12 Call the QOLSEND API with a X'B100' operation to clear the peer connection.

13 The source program will check its data queue for a response to the clear packet sent to the target system. Once the response is received, the program will clean up, call the QOLDLINK API to disable the link previously enabled, and end.

14 The following C functions illustrate the various user-defined communications support APIs.

15 This procedure illustrates a call to the QOLDLINK API. Note the vary option is set to vary off the associated AS/400 *USRDFN network device.

16 The settimer() calls the QOLTIMER API requesting timers for 20000 milliseconds, or twenty seconds. After setting a timer, the settimer() will call the dequeue() to remove an entry from the program's data queue.

17 The x25lind() illustrates calling the QOLQLIND API.

18 As mentioned in block **6**, the handler() will be called when OS/400 exception situation is encountered. This function performs final processing, calls the QOLDLINK API, and ends the source application program.

Target Application on System B Listing

The target application waits for the source application to initiate the file transfer. The following list identifies the actions of the target application:

- Calls the QOLQLIND API to get local X.25 line information
- Opens the local file
- Calls the QOLELINK API to establish a link for communications
- Calls the QOLSETF API to activate an X.25 protocol ID filter
- Calls the QOLRECV API to receive the X'B201' operation (incoming call)
- Calls the QOLSEND API with a X'B400' operation to accept the SVC connection
- Receives the file from the target system via X'0001' operations
- Calls the QOLRECV API to receive the X'B301' (connection failure notification)
- Call the QOLSEND API with 'B100' operation to locally close the SVC connection
- Calls the QOLDLINK API to disable the link
- Calls the QOLTIMER API to manage the reception of data queue entries

```
5738CX1 V2R1M0  910329   IBM SAA C/400         UDCS_APPLS/TARGET                    S1016897        12/19/90 09:01:25     Page      1
                                             * * * * *   P R O L O G   * * * * *
 Program name . . . . . . . . . :   TARGET
   Library name . . . . . . . . :      UDCS_APPLS
 Source file . . . . . . . . . . :   QCSRC
   Library name . . . . . . . . :      UDCS_APPLS
 Source member name  . . . . . . :   TARGET
 Text Description  . . . . . . . :  Target Applicatoin Example
 Compiler options  . . . . . . . :   *SOURCE     *NOXREF     *NOSHOWUSR  *NOSHOWSYS  *NOSHOWSKP  *NOEXPMAC   *NOAGR
                                 :   *NOPPONLY   *NODEBUG    *GEN        *NOSECLVL   *PRINT      *LOGMSG
 Language level options  . . . . :
 Source margins:
   Left margin . . . . . . . . . :   1
   Right margin  . . . . . . . . :   80
 Sequence columns:
   Left Column . . . . . . . . . :
   Right Column  . . . . . . . . :
 Define name . . . . . . . . . . :
 Generation options  . . . . . . :   *NOLIST     *NOXREF     *GEN        *NOATR      *NODUMP     *NOOPTIMIZE *NOALWBND
                                 :   *NOANNO
 Print file  . . . . . . . . . . :   QSYSPRT
   Library name  . . . . . . . . :      *LIBL
 Message flagging level  . . . . :   0
 Compiler message:
   Message limit . . . . . . . . :   *NOMAX
   Message limit severity  . . . :   30
 Replace program object  . . . . :   *YES
 User profile  . . . . . . . . . :   *USER
 Authority . . . . . . . . . . . :   *LIBCRTAUT
 Target Release  . . . . . . . . :   *CURRENT
 INDEBUG options . . . . . . . . :   I don't know
 Last change . . . . . . . . . . :   90/12/19 08:49:37
 Source description  . . . . . . :   Target Applicatoin Example
 Compiler  . . . . . . . . . . . :   IBM SAA C/400 Compiler
```

Figure 4-2 (Part 1 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400           UDCS_APPLS/TARGET                S1016897         12/19/90 09:01:25     Page     2
                                               * * * * *   S O U R C E   * * * * *
 Line  STMT                                                                                                            SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
            |/*******************************************************************/                          |
            |/**                                                            **/                          |
            |/**  Program Name: Target Application Program Example          **/                          |
            |/**                                                            **/                          |
            |/**                                                            **/                          |
            |/**  Function:                                                 **/                          |
            |/**  This is the target application program example that uses  **/                          |
            |/**  X.25 services provided by the user-defined communications  **/                          |
            |/**  support to receive a simple file from the source application **/                       |
            |/**  program running on System A.  This program performs the   **/                          |
            |/**  following:                                                **/                          |
            |/**     01.  Open the target file named OUTFILE.               **/                          |
            |/**     02.  Call QOLQLIND to obtain local line information.   **/                          |
            |/**     03.  Enable a link.                                    **/                          |
            |/**     04.  Set a Filter on the enabled link.                 **/                          |
            |/**     05.  Receive a 'B101'X operation (incoming call).      **/                          |
            |/**     06.  Send a 'B400'X operation (accept call).           **/                          |
            |/**     07.  Receive '0001'X operation(s) (incoming data) from **/                          |
            |/**          the source application program and write it to the **/                        |
            |/**          file opened in step 1).                           **/                          |
            |/**     08.  Receive a 'B301'X operation (clear call indication). **/                       |
            |/**     09.  Send a 'B100'X operation to respond locally to the **/                         |
            |/**          clearing of the connection.                       **/                          |
            |/**     10.  Disable the link enabled in step 3).              **/                          |
            |/**                                                            **/                          |
            |/**  A data queue will be actively used to manage the operation **/                         |
            |/**  of this program.  Data queue support will be used to monitor **/                       |
            |/**  for the completion of the enable and disable routines, as **/                          |
            |/**  well as timer expirations and incoming data.  Timers are  **/                          |
            |/**  used to ensure that there will never be an infinite wait on **/                        |
            |/**  the data queue.  If a timer expires, the link enabled will **/                         |
            |/**  be disabled and the program will stop.                    **/                          |
            |/**                                                            **/                          |
            |/**                                                            **/                          |
            |/** Inputs:                                                    **/                          |
            |/** The program expects the following input parameters:       **/                          |
            |/**      Line Name:   This is the name of the line description **/                         |
            |/**                   that will be used to call the QOLELINK API. **/                      |
            |/**                   The line must be an X.25 line with at least **/                      |
            |/**                   one SVC of type *SVCBOTH or *SVCIN.      **/                          |
            |/**                                                            **/                          |
            |/**      CommHandle:  This is the logical name that will be used **/                       |
            |/**                   to identify the link enabled.            **/                          |
            |/**                                                            **/                          |
            |/**      Remote DTE Address:  The is the Local Network Address **/                          |
            |/**                           of system A.                     **/                          |
            |/**                                                            **/                          |
            |/**                                                            **/                          |
            |/** Outputs:                                                   **/                          |
            |/** Current status of the file transfer will be provided when  **/                          |
            |/** running this program. If an error should occur, then a    **/                          |
            |/** message will be displayed indicating where the error occurred **/                       |
```

Figure 4-2 (Part 2 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                   S1016897         12/19/90 09:01:25    Page    3
 Line  STMT                                                                                                                SEQNBR  INCNO
        *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
       |/** and the program will end.        If the program completes     **/
       |/** successfully, a "successful completion" message will be       **/
       |/** posted.                                                       **/
       |/**                                                               **/
       |/*******************************************************************/
       |
       |#include "header"
       |
       |void senddata(sendparms *a, char *b, desc *c, char *d, char *e, int f);
       |
       |void sndformat1(sendparms *a,char *b, char *c, char *d, qlindparms *e);
       |
       |void sndformat2 (sendparms *a, char *b, char *c);
       |
       |void setfilters (hdrparms *a);
       |
       |void byte (char *a, int b, char *c, int d);
       |
       |void printespec (espec *a);
       |
       |void settimer(unsigned short *a,char *b,qentry *c,usrspace *d,char *e);
       |
       |void dequeue (int a, char *b, qentry *c, usrspace *d);
       |
       |void putdata (char *a, int b, FILE *c);
       |
       |void x25lind (qlindparms *a, char *b);
       |
       |void disablelink (disableparms *a, char *b, usrspace *c);
       |
       |void handler (disableparms a, usrspace *b);
       |
       |sigdata_t *sigdata(void);
       |
       |/**********************************************************/
       |/***************   Start Main Program   *******************/
       |/**********************************************************/
       |
       |main (int argc, char *argv??(??))
       |{
       |
       |/************  Variable  Declarations      *******************/
       |
       |  usrspace inbuff,       /*  Input Data Buffer */
       |           indesc,       /*  Input Buffer Descriptor  */
       |           outbuff,      /*  Output Data Buffer  */
       |           outdesc,      /*  Output Buffer Descriptor */
       |           qname;        /*  Data Queue  */
       |
       |  int length,                 /* Data Queue key legth  */
       |      inc, i, j;                      /*  counters  */
       |  unsigned short expctid;       /*  Message ID that is expected */
       |  char commhandle??(10??),     /*  Command Line Parameter  */
       |       rmtdte??(17??),         /*  Remote DTE Address  */
```

Figure 4-2 (Part 3 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET               S1016897        12/19/90 09:01:25     Page     4
 Line  STMT                                                                                                          SEQNBR   INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  107         |       *buffer,              /* Pointer to buffer     */                                     |    107
  108         |       key??(256??);         /* Data Queue key identifier  */                                |    108
  109         |  desc *descriptor;          /* Pointer to buffer descriptor  */                             |    109
  110         |                                                                                             |    110
  111         |/**  definitions for API functions  **/                                                      |    111
  112         |  enableparms enable;                                                                        |    112
  113         |  disableparms disable;                                                                      |    113
  114         |  sendparms send;                                                                            |    114
  115         |  recvparms recv;                                                                            |    115
  116         |  setfparms setf;                                                                            |    116
  117         |  timerparms timer;                                                                          |    117
  118         |  qlindparms qlind;                                                                          |    118
  119         |  qentry dataq;                                                                              |    119
  120         |  hdrparms *header;                                                                          |    120
  121         |                                                                                             |    121
  122         |  /******  Annnddddddd....  there off!!   ***********/                                       |    122
                 1
  123         |                                                                                             |    123
  124         |  /***---  Open the file to put the received data.     ----**/                               |    124
  125       1 |  if ((fptr = fopen("UDCS_APPLS/OUTFILE)", "w")) == NULL)                                    |    125
  126         |    {                                                                                        |    126
  127       2 |     printf("Unable to open target output file in UDCS_APPLS LIB.\n");                       |    127
  128       3 |     printf("The Program was terminated.\n\n");                                              |    128
  129       4 |     return;                                                                                 |    129
  130         |    }                                                                                        |    130
  131         |  /***---  Open the display file for error handling.   ----**/                               |    131
  132       5 |  if ((screen = fopen("ERRORSPEC", "ab+ type = record")) == NULL)                            |    132
  133         |    {                                                                                        |    133
  134       6 |     printf("Unable to open display file.\n");                                               |    134
  135       7 |     printf("The Program was terminated.\n\n");                                              |    135
  136       8 |     return;                                                                                 |    136
  137         |    }                                                                                        |    137
  138         |                                                                                             |    138
  139         |  /***---  Set the Execption Handler        ----**/                                          |    139
  140       9 |  signal(SIGABRT,&handler);                                                                  |    140
  141         |                                                                                             |    141
  142         |  /**  Clear the command line parameters  **/                                                |    142
  143      10 |  strncpy(enable.linename, "          ", 10);    /* Clear linename */                         |    143
  144      11 |  strncpy(commhandle, "          ", 10);     /* Clear Commhandle */                           |    144
  145      12 |  strncpy(rmtdte, "                 ", 17);     /* Clear Remote DTE */                         |    145
  146         |                                                                                             |    146
  147         |  /**  Receive command line Paramters  **/                                                   |    147
  148      13 |  strncpy(enable.linename, argv??(1??), strlen(argv??(1??)));                                |    148
  149      14 |  strncpy(commhandle, argv??(2??), strlen(argv??(2??)));                                     |    149
  150      15 |  strncpy(rmtdte, argv??(3??), strlen(argv??(3??)));                                         |    150
  151      16 |  rmtdte??(strlen(argv??(3??))??) = '\0';                                                    |    151
  152         |                                                                                             |    152
  153         |  /**   Initialize the user spaces  **/                                                      |    153
  154      17 |  strncpy(inbuff.library, "UDCS_APPLS", 10);    /*  Input Buffer */                          |    154
  155      18 |  strncpy(inbuff.name, "TARGETIBUF", 10);                                                    |    155
  156      19 |  strncpy(indesc.library, "UDCS_APPLS", 10);    /* Input B Desc  */                          |    156
  157      20 |  strncpy(indesc.name, "TARGETIDSC", 10);                                                    |    157
  158      21 |  strncpy(outbuff.library, "UDCS_APPLS", 10);   /*  Output Buffer*/                          |    158
  159      22 |  strncpy(outbuff.name, "TARGETOBUF", 10);                                                   |    159
  160      23 |  strncpy(outdesc.library, "UDCS_APPLS", 10);   /* Output B Desc */                          |    160
```

Figure 4-2 (Part 4 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                   S1016897         12/19/90 09:01:25      Page     5
 Line  STMT                                                                                                                SEQNBR  INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  161    24 | strncpy(outdesc.name, "TARGETODSC", 10);                                                        |    161
  162    25 | strncpy(qname.library, "UDCS_APPLS", 10);   /*  Data queue  */                                  |    162
  163    26 | strncpy(qname.name, "X25DTAQ   ", 10);                                                          |    163
  164       |                                                                                                 |    164
  165       | /*****   retrieve the line description information  ******/                                     |    165
  166    27 | x25lind (&qlind, enable.linename);                                                              |    166
  167       |                                                                                                 |    167
  168    28 | if ((qlind.retcode != 0) || (qlind.reason != 0))                                                |    168
  169       |    {                                                                                            |    169
  170    29 |    printf("Query line description failed.\n");                                                  |    170
  171    30 |    printf("Return code = %d\n", qlind.retcode);                                                 |    171
  172    31 |    printf("Reason code = %d\n\n", qlind.reason);                                                |    172
  173    32 |    return;                                                                                      |    173
  174       |    }                                                                                            |    174
  175       |                                                                                                 |    175
  176       | /*****   Hard Code the QOLELINK Input Parameters  ******/                                       |    176
  177    33 | enable.maxdtax25 = 512;                                                                         |    177
  178    34 | enable.keylength = 3;                                                                           |    178
  179    35 | strncpy(enable.keyvalue, "RCV", 3);                                                             |    179
            2
  180       |                                                                                                 |    180
  181       | /**-------  Enable the link  -----------**/                                                     |    181
  182       | QOLELINK (&(enable.retcode), &(enable.reason), &(enable.tdusize),\                              |    182
  183       |     &(enable.numunits), &(enable.maxdtalan), &(enable.maxdtax25),\                              |    183
  184       |     (char *)&inbuff, (char *)&indesc, (char *)&outbuff,\                                        |    184
  185       |     (char *)&outdesc, &(enable.keylength), enable.keyvalue,\                                    |    185
  186    36 |     (char *)&qname, enable.linename, commhandle);                                               |    186
  187       |                                                                                                 |    187
  188    37 | if ((enable.retcode != 0) || (enable.reason != 0))                                              |    188
  189       |    {                                                                                            |    189
  190       |    printf("Line %.10s with Commhandle %.10s was NOT ENABLED.\n",\                               |    190
  191    38 |            enable.linename, commhandle);                                                        |    191
  192    39 |    printf("Return code = %d\n", enable.retcode);                                                |    192
  193    40 |    printf("Reason code = %d\n\n", enable.reason);                                               |    193
  194    41 |    return;                                                                                      |    194
  195       |    }                                                                                            |    195
            3
  196       |                                                                                                 |    196
  197       | /**-------  Set a timer for Enable link  --------**/                                            |    197
  198    42 | expctid = 0xF0F0;                                                                               |    198
  199    43 | settimer(&expctid, "Enable", &dataq, &qname, commhandle);                                       |    199
  200    44 | if (expctid != 0xF0F0)                                                                          |    200
  201       |    {                                                                                            |    201
  202    45 |    disablelink (&disable, commhandle, &qname);                                                  |    202
  203    46 |    return;                                                                                      |    203
  204       |    }                                                                                            |    204
  205       |                                                                                                 |    205
  206       |                                                                                                 |    206
  207       | /************************************************************/                                  |    207
  208       | /*******----  Set a Filter for the Link  --------*********/                                      |    208
  209       | /************************************************************/                                  |    209
            4
  210       |                                                                                                 |    210
  211    47 | QUSPTRUS(&outbuff, &header);   /*  get the output buffer pointer */                             |    211
  212    48 | header->function = 1;          /*  add a filter   */                                            |    212
  213    49 | header->type = 0;              /*  X.25 PID only  */                                            |    213
  214    50 | header->number = 1;            /*  set 1 filter   */                                            |    214
```

Figure 4-2 (Part 5 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                 S1016897       12/19/90 09:01:25     Page    6
 Line  STMT                                                                                                         SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  215    51 | header->length = 16;           /*  X.25 filter length  */                                     |    215
  216    52 | setfilters(header);            /*  Fill in the filter format */                               |    216
  217       |                                                                                                |    217
  218       | /*******---- Set the filter for the Link --------*********/                                     |    218
  219       | QOLSETF (&(setf.retcode), &(setf.reason), &(setf.erroffset),\                                  |    219
  220    53 |               commhandle);                                                                     |    220
  221    54 | if ((setf.retcode != 0) || (setf.reason != 0))                                                 |    221
  222       |    {                                                                                           |    222
  223    55 |    printf("Set Filters Return Code = %.2d\n", setf.retcode);                                   |    223
  224    56 |    printf("Set Filters Reason Codes = %.4d\n", setf.reason);                                   |    224
  225    57 |    printf("Set Filters Error Offset = %.4d\n", setf.erroffset);                                |    225
  226    58 |    return;                                                                                     |    226
  227       |    }                                                                                           |    227
  228       |                                                                                                |    228
  229       |                                                                                                |    229
  230       | /************************************************************/                                |    230
  231       | /****  Receive the incoming call packet and accpet the call **/                                |    231
  232       | /************************************************************/                                |    232
  233       |                                                                                                |    233
  234       | /**-------  Set a timer to receive data  --------**/                                           |    234
  235    59 | expctid = 0xF0F3;                                                                              |    235
  236    60 | settimer(&expctid, "Inc Call ", &dataq, &qname, commhandle);                                   |    236
  237    61 | if (expctid != 0xF0F3)                                                                         |    237
  238       |    {                                                                                           |    238
  239    62 |    disablelink (&disable, commhandle, &qname);                                                 |    239
  240    63 |    return;                                                                                     |    240
  241       |    }                                                                                           |    241
              5
  242       |                                                                                                |    242
  243       | /**********  Receive the Incoming Data     **********/                                         |    243
  244    64 | QUSPTRUS (&inbuff, &buffer);                                                                   |    244
  245    65 | QUSPTRUS (&indesc, &descriptor);                                                               |    245
  246       |                                                                                                |    246
  247       | QOLRECV (&(recv.retcode), &(recv.reason), &(recv.ucep),\                                       |    247
  248       |          &(recv.pcep), &(recv.operation), &(recv.numdtaunits),\                                |    248
  249    66 |          &(recv.dataavail), &(recv.errorspecific), commhandle);                                |    249
  250       |                                                                                                |    250
  251    67 | if ((recv.retcode != 0) || (recv.reason != 0))                                                 |    251
  252       |    {                                                                                           |    252
  253    68 |    printf("Recv incoming call packet failed\n");                                               |    253
  254    69 |    printf("return code %d\n", recv.retcode);                                                   |    254
  255    70 |    printf("reason code %d\n", recv.reason);                                                    |    255
  256    71 |    printespec(&(send.errorspecific));                                                          |    256
  257       |                                                                                                |    257
  258    72 |    disablelink (&disable, commhandle, &qname);                                                 |    258
  259    73 |    return;                                                                                     |    259
  260       |    }                                                                                           |    260
  261       |                                                                                                |    261
  262       | /*** Interpret the Received Operation  ***/                                                    |    262
  263    74 | if (recv.operation != 0xB201)                                                                  |    263
  264       |    {                                                                                           |    264
  265    75 |    printf("Recvd operation %x instead of B201", recv.operation);                               |    265
  266    76 |    disablelink (&disable, commhandle, &qname);                                                 |    266
  267    77 |    return;         /****  End the program  ***/                                                |    267
  268       |    }                                                                                           |    268
```

Figure 4-2 (Part 6 of 19). C/400 Compiler Listing for the Target Application

```
      STMT                                                                                              SEQNBR  INCNO
           *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
          |                                                                                          |
          6
          |                                                                                          |
          |  /*********************************************************************/                 |
          |  /** Send a response to accept the call and establish a connection **/                   |
          |  /*********************************************************************/                 |
          |                                                                                          |
          |  /****   Get pointers to the user spaces.   ******/                                       |
       78 |  QUSPTRUS(&outbuff, &buffer);                                                            |
       79 |  QUSPTRUS(&outdesc, &descriptor);                                                        |
          |                                                                                          |
          |  /*******   Set up Send Packet    *********/                                             |
       80 |  send.ucep = 62;                    /*  set UCEP to be 62  */                            |
       81 |  send.pcep = recv.pcep;             /*  get the PCEP number  */                          |
       82 |  send.operation = 0xB400;           /*  send a call request response */                  |
       83 |  send.numdtaelmnts = 1;             /*  send one data unit   */                          |
          |                                                                                          |
          |  /**-----   Send the packet   ----------------**/                                        |
       84 |  sndformat1 (&send, buffer, rmtdte, commhandle, &qlind);                                 |
          |                                                                                          |
       85 |  if ((send.retcode != 0) || (send.reason != 0))                                          |
          |     {                                                                                    |
       86 |     printf("Data NOT sent for commhandle %.9s\n", commhandle);                           |
       87 |     printf("Return code = %d\n", send.retcode);                                          |
       88 |     printf("Reason code = %d\n", send.reason);                                           |
       89 |     printf("new pcep %d\n\n", send.newpcep);                                             |
       90 |     printespec(&(send.errorspecific));                                                   |
          |                                                                                          |
       91 |     disablelink (&disable, commhandle, &qname);                                          |
       92 |     return;                                                                              |
          |     }                                                                                    |
          |                                                                                          |
       93 |  printf("An X.25 SVC connection was completed\n\n");                                     |
          |                                                                                          |
          7
          |                                                                                          |
          |  /***********************************************************/                           |
          |  /****    Receive Incoming Data    *************************/                           |
          |  /***********************************************************/                           |
          |                                                                                          |
          |  /**-------  Set a timer to receive data  --------**/                                    |
       94 |  expctid = 0xF0F3;                                                                       |
       95 |  settimer(&expctid, "Inc Data ", &dataq, &qname, commhandle);                            |
       96 |  if (expctid != 0xF0F3)                                                                  |
          |     {                                                                                    |
       97 |     disablelink (&disable, commhandle, &qname);                                          |
       98 |     return;                                                                              |
          |     }                                                                                    |
          |                                                                                          |
          |  /*******---  Receive the Incoming Data     ----******/                                   |
          |  /** Get pointer to user space  **/                                                      |
       99 |  QUSPTRUS (&inbuff, &buffer);                                                            |
      100 |  QUSPTRUS (&indesc, &descriptor);                                                        |
          |                                                                                          |
          |  /**  Receive the data   **/                                                            |
          |  QOLRECV (&(recv.retcode), &(recv.reason), &(recv.ucep),\                               |
```

Figure 4-2 (Part 7 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400           UDCS_APPLS/TARGET              S1016897        12/19/90 09:01:25     Page     8
 Line  STMT                                                                                                         SEQNBR  INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  323       |          &(recv.pcep), &(recv.operation), &(recv.numdtaunits),\                             |    323
  324   101 |          &(recv.dataavail), &(recv.errorspecific), commhandle);                             |    324
  325       |                                                                                             |    325
  326   102 |  if ((recv.retcode != 0) || (recv.reason != 0))                                             |    326
  327       |     {                                                                                       |    327
  328   103 |     printf("Recv op for first data unit failed\n");                                         |    328
  329   104 |     printf("return code %d\n", recv.retcode);                                               |    329
  330   105 |     printf("reason code %d\n", recv.reason);                                                |    330
  331   106 |     printespec(&(send.errorspecific));                                                      |    331
  332       |                                                                                             |    332
  333   107 |     disablelink (&disable, commhandle, &qname);                                             |    333
  334   108 |     return;                                                                                 |    334
  335       |     }                                                                                       |    335
  336       |                                                                                             |    336
              8
  337       |                                                                                             |    337
  338       |  /****************************************************************/                         |    338
  339       |  /*******   Start a loop to read in all the incoming data      ***/                          |    339
  340       |  /****************************************************************/                         |    340
  341       |                                                                                             |    341
  342   109 |  i = 1;                                                                                     |    342
  343   110 |  while (recv.operation == 0x0001)                                                           |    343
  344       |     {                                                                                       |    344
  345   111 |     printf("%d Data Recvd {%.4x}.\n\n", i++, recv.operation);                               |    345
  346       |                                                                                             |    346
  347       |                                                                                             |    347
  348       |     /**  Store all the data units in the file  **/                                          |    348
  349   112 |     for (j = 1; j <= recv.numdtaunits; j++) {                                               |    349
  350       |        putdata (buffer + (j - 1)*enable.tdusize,\                                           |    350
  351   113 |                 descriptor->length, fptr);                                                  |    351
  352   114 |        descriptor = (desc *)((char *)descriptor + sizeof(desc));                            |    352
  353       |        }   /* for */                                                                        |    353
  354       |                                                                                             |    354
  355       |     /**-------  Set a timer to wait for more data  -------**/                               |    355
  356   115 |     if (recv.dataavail == 0)                                                                |    356
  357       |        {                                                                                    |    357
  358       |        /** Set timer  **/                                                                   |    358
  359   116 |        expctid = 0xF0F3;                                                                    |    359
  360   117 |        settimer(&expctid, "Wt Inc Dta", &dataq, &qname, commhandle);                        |    360
  361   118 |        if (expctid != 0xF0F3)                                                               |    361
  362       |           {                                                                                 |    362
  363   119 |           disablelink (&disable, commhandle, &qname);                                       |    363
  364   120 |           return;                                                                           |    364
  365       |           }                                                                                 |    365
  366       |        }                                                                                    |    366
  367       |                                                                                             |    367
  368       |     /** Get pointer to user space  **/                                                      |    368
  369   121 |     QUSPTRUS (&inbuff, &buffer);                                                            |    369
  370   122 |     QUSPTRUS (&indesc, &descriptor);                                                        |    370
  371       |                                                                                             |    371
  372       |     /**  Receive the data   **/                                                             |    372
  373       |     QOLRECV (&(recv.retcode), &(recv.reason), &(recv.ucep),\                                |    373
  374       |             &(recv.pcep), &(recv.operation), &(recv.numdtaunits),\                          |    374
  375   123 |             &(recv.dataavail), &(recv.errorspecific), commhandle);                          |    375
  376       |                                                                                             |    376
  377       |     }    /**  End  Receive data  while loop   ******/                                       |    377
  378       |                                                                                             |    378
```

Figure 4-2 (Part 8 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                  S1016897        12/19/90 09:01:25     Page    9
 Line  STMT                                                                                                             SEQNBR  INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
             9
  379        |                                                                                                   |   379
  380        |  /*****************************************************/                                         |   380
  381        |  /*********** Receive the Clear indication ***********/                                         |   381
  382        |  /*****************************************************/                                         |   382
  383        |                                                                                                   |   383
  384   124  |  if ((recv.retcode != 83) || (recv.reason != 4002))                                               |   384
  385        |     {                                                                                             |   385
  386   125  |     printf("Recv opr for clear request failed\n");                                                |   386
  387   126  |     printf("return code %d\n", recv.retcode);                                                     |   387
  388   127  |     printf("reason code %d\n", recv.reason);                                                      |   388
  389   128  |     printespec(&(send.errorspecific));                                                            |   389
  390        |                                                                                                   |   390
  391   129  |     disablelink (&disable, commhandle, &qname);                                                   |   391
  392   130  |     return;                                                                                       |   392
  393        |     }                                                                                             |   393
  394        |                                                                                                   |   394
  395        |  /* Interpret the Received Operation  */                                                         |   395
  396   131  |  if (recv.operation != 0xB301)                                                                    |   396
  397        |     {                                                                                             |   397
  398   132  |     printf("Recvd operation %x instead of B301", recv.operation);                                 |   398
  399   133  |     disablelink (&disable, commhandle, &qname);                                                   |   399
  400   134  |     return;         /****  end the program  ***/                                                  |   400
  401        |     }                                                                                             |   401
  402        |                                                                                                   |   402
              10
  403        |                                                                                                   |   403
  404        |  /****************************************************************/                              |   404
  405        |  /***********  Send local response to clear indication ***********/                              |   405
  406        |  /****************************************************************/                              |   406
  407        |                                                                                                   |   407
  408        |  /****   Get pointers to the user spaces.   ******/                                              |   408
  409   135  |  QUSPTRUS(&outbuff, &buffer);                                                                     |   409
  410   136  |  QUSPTRUS(&outdesc, &descriptor);                                                                 |   410
  411        |                                                                                                   |   411
  412        |  /*******   Set up the packet         ****************/                                          |   412
  413   137  |  send.operation = 0xB100;       /*  send a clear request packet  */                               |   413
  414   138  |  send.numdtaelmnts = 1;         /*  send one data unit  */                                        |   414
  415        |                                                                                                   |   415
  416        |  /**-----   Send the packet   ----------------**/                                                 |   416
  417   139  |  sndformat2 (&send, buffer, commhandle);                                                          |   417
  418        |                                                                                                   |   418
  419   140  |  if ((send.retcode != 0) && (send.reason != 0))                                                   |   419
  420        |     {                                                                                             |   420
  421   141  |     printf("Response not sent for clear connection\n");                                           |   421
  422   142  |     printf("Return code = %d\n", send.retcode);                                                   |   422
  423   143  |     printf("Reason code = %d\n", send.reason);                                                    |   423
  424   144  |     printf("new pcep %d\n\n", send.newpcep);                                                      |   424
  425   145  |     printespec(&(send.errorspecific));                                                            |   425
  426        |                                                                                                   |   426
  427   146  |     disablelink (&disable, commhandle, &qname);                                                   |   427
  428   147  |     return;                                                                                       |   428
  429        |     }                                                                                             |   429
  430        |                                                                                                   |   430
  431        |                                                                                                   |   431
  432        |  /******************************************************/                                        |   432
```

Figure 4-2 (Part 9 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET               S1016897        12/19/90 09:01:25     Page    10
 Line  STMT                                                                                                             SEQNBR   INCNO
              *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  433       |  /*********** Receive the Clear Confirmation  **********/                                     |    433
  434       |  /******************************************************/                                     |    434
  435       |                                                                                               |    435
  436       |  /**-------  Set a timer to receive data  --------**/                                         |    436
  437   148 |  expctid = 0xF0F3;                                                                            |    437
  438   149 |  settimer(&expctid, "Clr Cnfrm", &dataq, &qname, commhandle);                                 |    438
  439   150 |  if (expctid != 0xF0F3)                                                                       |    439
  440       |     {                                                                                         |    440
  441   151 |     disablelink (&disable, commhandle, &qname);                                               |    441
  442   152 |     return;                                                                                   |    442
  443       |     }                                                                                         |    443
  444       |                                                                                               |    444
  445   153 |  if ((recv.retcode != 00) || (recv.reason != 0000))                                           |    445
  446       |     {                                                                                         |    446
  447   154 |     printf("Recv failed for clear confirmation\n");                                           |    447
  448   155 |     printf("return code %d\n", recv.retcode);                                                 |    448
  449   156 |     printf("reason code %d\n", recv.reason);                                                  |    449
  450   157 |     printespec(&(send.errorspecific));                                                        |    450
  451       |                                                                                               |    451
  452   158 |     disablelink (&disable, commhandle, &qname);                                               |    452
  453   159 |     return;                                                                                   |    453
  454       |     }                                                                                         |    454
  455       |                                                                                               |    455
  456       |  /* Interpret the Received Operation  */                                                      |    456
  457   160 |  if (recv.operation != 0xB101)                                                                |    457
  458       |     {                                                                                         |    458
  459   161 |     printf("Recvd opr %x instead of opr B301\n", recv.operation);                             |    459
  460   162 |     disablelink (&disable, commhandle, &qname);                                               |    460
  461   163 |     return;                                                                                   |    461
  462       |     }                                                                                         |    462
              11
  463       |                                                                                               |    463
  464       |  /****************************************/                                                   |    464
  465       |  /**  disable the link and end program  **/                                                   |    465
  466       |  /****************************************/                                                   |    466
  467       |                                                                                               |    467
  468   164 |  disablelink (&disable, commhandle, &qname);                                                  |    468
  469       |                                                                                               |    469
  470       |                                                                                               |    470
  471   165 |  printf("TARGET application completed OK!\n\n");                                              |    471
  472       |                                                                                               |    472
  473       |}    /* End Main */                                                                            |    473
  474       |                                                                                               |    474
  475       |                                                                                               |    475
  476       |                                                                                               |    476
  477       |/*****************************************************************/                            |    477
  478       |/************** Start Subroutine Section    **********************/                            |    478
  479       |/*****************************************************************/                            |    479
  480       |                                                                                               |    480
  481       |/*****************************************************************/                            |    481
  482       |/************  Routine to fill X.25 Format I   ***********/                                     |    482
  483       |                                                                                               |    483
  484       |void sndformat1 (sendparms *send,                                                              |    484
  485       |                 char *buffer,                                                                 |    485
  486       |                 char *rmtdte,                                                                 |    486
```

Figure 4-2 (Part 10 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                    S1016897        12/19/90 09:01:25     Page   11
 Line  STMT                                                                                                             SEQNBR  INCNO
           *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  487       |               char *commhandle,                                                                  |   487
  488       |               qlindparms *qlind)                                                                 |   488
  489       |                                                                                                  |   489
  490       |{                                                                                                 |   490
  491       |  format1 *output = (format1 *) buffer;                                                           |   491
  492       |  register int counter;                                                                           |   492
  493       |  register querydata *qd;                                                                         |   493
  494       |                                                                                                  |   494
  495     1 |  qd = (querydata *)&(qlind->userbuffer);                                                         |   495
  496     2 |  output->type = 0;                 /*  not used  */                                              |   496
  497     3 |  output->logchanid = 0x0;                                                                        |   497
  498     4 |  output->sendpacksize = qd->x25data.defsend;                                                     |   498
  499     5 |  output->sendwindsize = qd->x25data.windowsend;                                                  |   499
  500     6 |  output->recvpacksize = qd->x25data.defrecv;                                                     |   500
  501     7 |  output->recvwindsize = qd->x25data.windowrecv;                                                  |   501
  502       |                                                                                                  |   502
  503     8 |  output->dtelength = strlen(rmtdte);               /*  not used  */                              |   503
  504     9 |  byte(output->dte, 16, rmtdte, strlen(rmtdte));    /*  not used  */                              |   504
  505    10 |  output->dbit = 0;                                                                               |   505
  506    11 |  output->cug = 0;                                  /*  not used  */                              |   506
  507    12 |  output->cugid = 0;                                /*  not used  */                              |   507
  508    13 |  output->reverse = 0;                              /*  not used  */                              |   508
  509    14 |  output->fast = 0;                                 /*  not used  */                              |   509
  510    15 |  output->faclength = 0;                                                                          |   510
  511    16 |  byte(output->facilities, 109, "", 0);                                                           |   511
  512    17 |  output->calllength = 0;                                                                         |   512
  513    18 |  byte(output->callud, 128, "00", 2);                                                             |   513
  514    19 |  output->misc??(0??) = 0;                                                                        |   514
  515    20 |  output->misc??(1??) = 0;                                                                        |   515
  516    21 |  output->misc??(2??) = 0;                                                                        |   516
  517    22 |  output->misc??(3??) = 0;                                                                        |   517
  518    23 |  output->maxasmsize = 16383;                                                                     |   518
  519    24 |  output->autoflow = 32;                                                                          |   519
  520       |                                                                                                  |   520
  521       |                                                                                                  |   521
  522       |  QOLSEND (&(send->retcode), &(send->reason), &(send->errorspecific),\                            |   522
  523       |           &(send->newpcep), &(send->ucep), &(send->pcep),\                                       |   523
  524    25 |           commhandle, &(send->operation), &(send->numdtaelmnts));                                |   524
  525       |                                                                                                  |   525
  526       |}  /*  End sndformat1 Subroutine  */                                                              |   526
  527       |                                                                                                  |   527
  528       |                                                                                                  |   528
  529       |/*****************************************************************/                               |   529
  530       |/************  Routine to fill X.25 Format II   ***********/                                       |   530
  531       |                                                                                                  |   531
  532       |void sndformat2 (sendparms *send,                                                                 |   532
  533       |               char *buffer,                                                                      |   533
  534       |               char *commhandle)                                                                  |   534
  535       |                                                                                                  |   535
  536       |{                                                                                                 |   536
  537       |  format2 *output = (format2 *) buffer;                                                           |   537
  538       |                                                                                                  |   538
  539     1 |  output->type = 1;                                                                               |   539
  540     2 |  output->cause = 'FF';                                                                           |   540
```

Figure 4-2 (Part 11 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                S1016897        12/19/90 09:01:25     Page    12
 Line  STMT                                                                                                        SEQNBR   INCNO
           *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  541      3 |  output->diagnostic = 'FF';                                                               |    541
  542      4 |  output->faclength = 0;                                                                   |    542
  543      5 |  byte(output->facilities, 109, "", 0);                                                    |    543
  544      6 |  output->length = 0;                                                                      |    544
  545      7 |  byte(output->userdata, 128, "", 0);                                                      |    545
  546        |                                                                                           |    546
  547        |                                                                                           |    547
  548        |  QOLSEND (&(send->retcode), &(send->reason), &(send->errorspecific),\                     |    548
  549        |           &(send->newpcep), &(send->ucep), &(send->pcep),\                                |    549
  550      8 |           commhandle, &(send->operation), &(send->numdtaelmnts));                         |    550
  551        |                                                                                           |    551
  552        |}  /*  End sndformat2 Subroutine  */                                                       |    552
  553        |                                                                                           |    553
  554        |                                                                                           |    554
  555        |/*********************************************************************/                   |    555
  556        |/**************   Fill in the Buffer for the Filter  ****************/                     |    556
  557        |                                                                                           |    557
  558        |void setfilters (hdrparms *header)                                                         |    558
  559        |                                                                                           |    559
  560        |{                                                                                          |    560
  561        |  x25filter *filters;                                                                      |    561
  562        |                                                                                           |    562
  563      1 |  filters = (x25filter *)header->filters;                                                  |    563
  564      2 |  filters??(0??).pidlength = 1;                                                            |    564
  565      3 |  filters??(0??).pid = 0x21;              /*  set the protocal ID  */                      |    565
  566      4 |  filters??(0??).dtelength = 0;           /* no DTE used in filter */                      |    566
  567      5 |  byte(filters??(0??).dte, 12, "", 0);                                                     |    567
  568      6 |  filters??(0??).flags = 0x0;                                                              |    568
  569      7 |  filters??(0??).flags += 0x80;        /* Set Reverse Charging to no  */                   |    569
  570      8 |  filters??(0??).flags += 0x40;        /*  Set Fast Select to no  */                       |    570
  571        |                                                                                           |    571
  572        |}  /*  End setfilters Subroutine  */                                                       |    572
  573        |                                                                                           |    573
  574        |                                                                                           |    574
  575        |/*********************************************/                                            |    575
  576        |/********    Routine to disable    ***********/                                            |    576
  577        |                                                                                           |    577
  578        |void disablelink (disableparms *disable,                                                   |    578
  579        |                  char *commhandle,                                                        |    579
  580        |                  usrspace *qname)                                                         |    580
  581        |                                                                                           |    581
  582        |{                                                                                          |    582
  583        |  qentry dataq;                                                                            |    583
  584        |  unsigned short expctid;                                                                  |    584
  585        |                                                                                           |    585
  586      1 |  disable->vary = 1;     /*  Hard code device to vary off  */                              |    586
  587        |                                                                                           |    587
  588        |  /**  Call disable link  **/                                                              |    588
  589        |  QOLDLINK (&(disable->retcode), &(disable->reason),\                                      |    589
  590      2 |                        commhandle, &(disable->vary));                                     |    590
  591        |                                                                                           |    591
  592      3 |  if ((disable->retcode != 0) && (disable->reason != 00))                                  |    592
  593        |     {                                                                                     |    593
  594      4 |     printf ("Link %.10s did not disabled.\n", commhandle);                                |    594
```

Figure 4-2 (Part 12 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                S1016897          12/19/90 09:01:25     Page    13
 Line  STMT                                                                                                    SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  595     5 |     printf ("return code = %d\n", disable->retcode);                                           |   595
  596     6 |     printf ("reason code = %d\n\n", disable->reason);                                          |   596
  597       |     }                                                                                           |   597
  598       |  else                                                                                           |   598
  599     7 |     printf ("%.10s link disabled \n", commhandle);                                             |   599
  600       |                                                                                                 |   500
  601       |  /**-------  Set a timer to receive message  --------**/                                      |   501
  602     8 |  expctid = 0xF0F1;                                                                              |   602
  603     9 |  settimer(&expctid, "Disable  ", &dataq, qname, commhandle);                                   |   603
  604    10 |  if (expctid != 0xF0F1)                                                                        |   604
  605       |     {                                                                                           |   605
  606    11 |     printf("Disable link did not complete successfully");                                      |   606
  607    12 |     return;                                                                                     |   607
  608       |     }                                                                                           |   608
  609       |                                                                                                 |   609
  610       |  /**  close the files  **/                                                                     |   610
  611    13 |  fclose(fptr);                                                                                  |   611
  612    14 |  fclose(screen);                                                                                |   612
  613       |                                                                                                 |   613
  614       |}  /*  End disablelink Subroutine  */                                                          |   614
  615       |                                                                                                 |   615
  616       |                                                                                                 |   616
  617       |/**************************************************************/                              |   617
  618       |/**    Routine to convert string to Hexidecimal format   ******/                                |   618
  619       |                                                                                                 |   619
  620       |void byte (char *dest,                                                                           |   620
  621       |           int dlength,                                                                          |   621
  622       |           char *source,                                                                         |   622
  623       |           int slength)                                                                          |   623
  624       |                                                                                                 |   624
  625       |{                                                                                                |   625
  626       |  register int counter;                                                                          |   626
  627       |  char holder??(2??);                                                                            |   627
  628       |                                                                                                 |   628
  629     1 |  for (counter=0;counter<dlength;counter++)                                                      |   629
  630     2 |   dest??(counter??)=0;                                                                          |   630
  631     3 |  for (counter=slength-1;counter>=0;counter--)                                                   |   631
  632     4 |   if isxdigit(source??(counter??))                                                             |   632
  633       |     {                                                                                           |   633
  634     5 |      holder??(0??)=source??(counter??);                                                        |   634
  635     6 |      holder??(1??)='\0';                                                                        |   635
  636     7 |      if  (counter % 2 == 0)                                                                     |   636
  637     8 |        dest??(counter/2??) += (char) hextoint(holder)*16;                                      |   637
  638     9 |      else dest??(counter/2??) += (char) hextoint(holder);                                      |   638
  639       |     }                                                                                           |   639
  640       |                                                                                                 |   640
  641       |}  /*  End byte Subroutine  */                                                                 |   641
  642       |                                                                                                 |   642
  643       |                                                                                                 |   643
  644       |/**************************************************************/                              |   644
  645       |/**    Routine to display the ErrorSpecific output       ******/                                |   645
  646       |/**************************************************************/                              |   646
  647       |                                                                                                 |   647
  648       |void printespec(espec *errorspecific)                                                            |   648
```

Figure 4-2 (Part 13 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400           UDCS_APPLS/TARGET              S1016897        12/19/90 09:01:25      Page    14
 Line  STMT                                                                                                            SEQNBR   INCNO
              *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  649        |                                                                                                  |    649
  650        |{                                                                                                 |    650
  651        |  especout outparms;                                                                              |    651
  652        |                                                                                                  |    652
  653      1 |  QXXFORMAT(screen, "ERRORSPEC ");                                                                |    653
  654      2 |  sprintf(outparms.hwecode, "%.8X", errorspecific->hwecode);                                      |    654
  655        |  sprintf(outparms.timestamp, "%.8X%.8X", errorspecific->timestamphi,\                            |    655
  656      3 |                                        errorspecific->timestamplo);                               |    656
  657      4 |  sprintf(outparms.elogid, "%.8X", errorspecific->elogid);                                        |    657
  658      5 |  if (errorspecific->flags & 0x40)                                                                |    658
  659      6 |    outparms.fail = 'Y';                                                                          |    659
  660      7 |  else outparms.fail = 'N';                                                                       |    660
  661      8 |  if (errorspecific->flags & 0x20)                                                                |    661
  662      9 |    outparms.zerocodes = 'Y';                                                                     |    662
  663     10 |  else outparms.zerocodes = 'N';                                                                  |    663
  664     11 |  if (errorspecific->flags & 0x10)                                                                |    664
  665     12 |    outparms.qsysopr = 'Y';                                                                       |    665
  666     13 |  else outparms.qsysopr = 'N';                                                                    |    666
  667     14 |  sprintf(outparms.cause,"%.2X", errorspecific->cause);                                           |    667
  668     15 |  sprintf(outparms.diagnostic, "%.2X", errorspecific->diagnostic);                                |    668
  669     16 |  sprintf(outparms.erroffset, "%.6d", errorspecific->erroroffset);                                |    669
  670     17 |  fwrite(&outparms, 1, sizeof(especout), screen);                                                 |    670
  671     18 |  fread("", 0, 0, screen);                                                                        |    671
  672        |                                                                                                  |    672
  673        |}  /*  End printespec Subroutine  */                                                              |    673
  674        |                                                                                                  |    674
  675        |                                                                                                  |    675
  676        |/*****************************************************************/                               |    676
  677        |/*******   Dequeues the Incoming Message and processes it ******/                                  |    677
  678        |                                                                                                  |    678
  679        |void dequeue (int length,                                                                         |    679
  680        |             char *key,                                                                           |    680
  681        |             qentry *dataq,                                                                       |    681
  682        |             usrspace *qname)                                                                     |    682
  683        |                                                                                                  |    683
  684        |{                                                                                                 |    684
  685        |  char fldlen??(3??),                                                                             |    685
  686        |       waittime??(3??),                                                                           |    686
  687        |       keylen??(2??),                                                                             |    687
  688        |       senderid??(2??),                                                                           |    688
  689        |       *pointer,                                                                                  |    689
  690        |       order??(2??);                                                                              |    690
  691        |  register int counter;                                                                           |    691
  692        |                                                                                                  |    692
  693        |                                                                                                  |    693
  694      1 |  waittime??(0??) = 0;                                                                            |    694
  695      2 |  waittime??(1??) = 0;                                                                            |    695
  696      3 |  waittime??(2??) = 0x1D;   /*  Hard code a delay of infinite   */                               |    696
  697      4 |  keylen??(0??) = 0;                                                                              |    697
  698      5 |  keylen??(1??) = 0x6F;  /* Hard code a keylength of 6  */                                        |    698
  699      6 |  senderid??(0??) = 0;                                                                            |    699
  700      7 |  senderid??(1??) = 0x0F;                                                                         |    700
  701      8 |  strncpy(order, "EQ", 2);                                                                        |    701
  702        |                                                                                                  |    702
```

Figure 4-2 (Part 14 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400           UDCS_APPLS/TARGET                  S1016897        12/19/90 09:01:25     Page    15
 Line  STMT                                                                                                   SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  703       |  /*  Clear the data structures  **/                                                         |    703
  704     9 |  fflush(stdin);                                                                             |    704
  705    10 |  pointer = (char *)dataq;                                                                   |    705
  706    11 |  for (counter = 0; counter < 336; counter++)                                                |    706
  707    12 |    pointer??(counter??) = 0;                                                                |    707
  708       |                                                                                             |    708
  709    13 |  strncpy (dataq->type, "       ", 7);                                                      |    709
  710    14 |  while ((strncmp(dataq->type, "*USRDFN", 7) != 0) || (fldlen == 0))                        |    710
  711       |     QRCVDTAQ(qname->name, qname->library, fldlen, dataq, waittime,\                        |    711
  712    15 |           order, keylen, key, senderid,"");                                                 |    712
  713       |                                                                                             |    713
  714       |}  /*  End dequeue Subroutine  */                                                           |    714
  715       |                                                                                             |    715
  716       |                                                                                             |    716
  717       |/**********************************************************/                               |    717
  718       |/*************   Set a timer and dequeue next entry  ******/                                 |    718
  719       |                                                                                             |    719
  720       |void settimer (unsigned short *expctid,                                                      |    720
  721       |               char *process,                                                                |    721
  722       |               qentry *dataq,                                                                |    722
  723       |               usrspace *qname,                                                              |    723
  724       |               char *commhandle)                                                             |    724
  725       |                                                                                             |    725
  726       |{                                                                                            |    726
  727       |timerparms timer;                                                                            |    727
  728       |disableparms disable;                                                                        |    728
  729       |int length;                                                                                  |    729
  730       |char key??(6??);                                                                             |    730
  731       |                                                                                             |    731
  732     1 |  timer.interval = 20000;             /*  set timer for 20 seconds */                        |    732
  733     2 |  timer.establishcount = 1;            /*  set establish count to 1 */                       |    733
  734     3 |  timer.keylength = 6;                 /*  key value  */                                     |    734
  735     4 |  strncpy(timer.keyvalue, "TARGET", 6);   /*  set key value /                                |    735
  736     5 |  timer.operation = 1;                 /*  set a timer  */                                   |    736
  737       |                                                                                             |    737
  738       |  /* Call QOLTIMER   */                                                                       |    738
  739       |  QOLTIMER (&(timer.retcode), &(timer.reason), timer.handleout,\                             |    739
  740       |           timer.handlein, (char *)qname, &(timer.operation),\                               |    740
  741       |           &(timer.interval), &(timer.establishcount),\                                      |    741
  742     6 |           &(timer.keylength), timer.keyvalue, timer.userdata);                              |    742
  743       |                                                                                             |    743
  744     7 |  if ((timer.retcode != 0) || (timer.reason != 0))                                           |    744
  745       |     {                                                                                       |    745
  746     8 |     printf("%s timer failed while being set.\n", process);                                  |    746
  747     9 |     printf("Return code = %d\n", timer.retcode);                                            |    747
  748    10 |     printf("Reason code = %d\n\n", timer.reason);                                           |    748
  749       |     }                                                                                       |    749
  750       |                                                                                             |    750
  751       |  /**-------  Dequeue an entry  --------**/                                                   |    751
  752    11 |  strncpy(key, "TARGET", 6);                                                                 |    752
  753    12 |  length = 6;                                                                                |    753
  754    13 |  dequeue (length, key, dataq, qname);                                                       |    754
  755       |                                                                                             |    755
  756       |  /***----     Cancel timer   -----***/                                                      |    756
```

Figure 4-2 (Part 15 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                S1016897        12/19/90 09:01:25      Page    16
 Line  STMT                                                                                                      SEQNBR   INCNO
              *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
  757     14 |  if (dataq->msgid != 0xF0F4)                                                                 |   757
  758        |     {                                                                                        |   758
  759     15 |     strncpy(timer.handlein, timer.handleout, 8);                                             |   759
  760     16 |     timer.operation = 2;           /*  Cancel one timer  */                                  |   760
  761        |                                                                                              |   761
  762        |     QOLTIMER (&(timer.retcode), &(timer.reason), timer.handleout,\                           |   762
  763        |           timer.handlein, (char *)qname, &(timer.operation),\                                |   763
  764        |           &(timer.interval), &(timer.establishcount),\                                       |   764
  765     17 |           &(timer.keylength), timer.keyvalue, timer.userdata);                               |   765
  766        |                                                                                              |   766
  767     18 |     if ((timer.retcode != 0) || (timer.reason != 0))                                         |   767
  768        |        {                                                                                     |   768
  769     19 |        printf("%s timer failed while being canceled\n", process);                            |   769
  770     20 |        printf("Return code = %d\n", timer.retcode);                                          |   770
  771     21 |        printf("Reason code = %d\n\n", timer.reason);                                         |   771
  772        |        }                                                                                     |   772
  773        |     }                                                                                        |   773
  774        |                                                                                              |   774
  775     22 |  if (dataq->msgid != *expctid)                                                               |   775
  776        |     {                                                                                        |   776
  777        |     printf ("A %.4X message ID was received instead of %.4X\n",\                             |   777
  778     23 |             dataq->msgid, *expctid);                                                         |   778
  779     24 |     printf ("%s completion message was not received\n", process);                            |   779
  780     25 |     *expctid = dataq->msgid;                                                                 |   780
  781        |     }                                                                                        |   781
  782        |                                                                                              |   782
  783        |}  /*  End settimer Subroutine  */                                                            |   783
  784        |                                                                                              |   784
  785        |                                                                                              |   785
  786        |/************************************************************/                                |   786
  787        |/**  x25lind:  Read a record into buf and return length    **/                                |   787
  788        |                                                                                              |   788
  789        |void x25lind (qlindparms *qlind, char *linename)                                              |   789
  790        |{                                                                                             |   790
  791        |register int counter;                                                                         |   791
  792        |                                                                                              |   792
  793      1 |  for(counter=0;counter<256;counter++)                                                        |   793
  794      2 |     qlind->userbuffer??(counter??)=0;                                                        |   794
  795        |                                                                                              |   795
  796      3 |  qlind->format = 0x01;                                                                       |   796
  797        |  QOLQLIND (&(qlind->retcode), &(qlind->reason), &(qlind->nbytes),\                           |   797
  798      4 |            qlind->userbuffer, linename, &(qlind->format));                                   |   798
  799        |                                                                                              |   799
  800        |}  /*  End x25lind Subroutine  */                                                             |   800
  801        |                                                                                              |   801
  802        |/************************************************************/                                |   802
  803        |/**  putdata:  Read a record into buf and return length    **/                                |   803
  804        |                                                                                              |   804
  805        |void putdata (char *buf,                                                                      |   805
  806        |              int dtalen,                                                                     |   806
  807        |              FILE *fptr)                                                                     |   807
  808        |                                                                                              |   808
  809        |{                                                                                             |   809
  810        |int i;                                                                                        |   810
```

Figure 4-2 (Part 16 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                 S1016897        12/19/90 09:01:25     Page    17
 Line  STMT                                                                                                         SEQNBR   INCNO
             *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  811       |                                                                                                 |     811
  812     1 |   for (i = 0; i < dtalen; i++)                                                                  |     812
  813     2 |      fwrite(buf + i, 1, 1, fptr);                                                               |     813
  814       |                                                                                                 |     814
  815       |}  /*  End putdata Subroutine  */                                                                |     815
  816       |                                                                                                 |     816
  817       |                                                                                                 |     817
  818       |/*  Exception handler, so that a failure will not                                                |     818
  819       |    kill the program, and any associated data!!  */                                              |     819
  820       |                                                                                                 |     820
  821       |void handler(disableparms disable, usrspace *qname)                                              |     821
  822       |{                                                                                                |     822
  823       |  sigdata_t *data;                                                                               |     823
  824       |                                                                                                 |     824
  825     1 |  disablelink(&disable, "*ALL      ", qname);                                                     |     825
  826     2 |  printf("The program received an excecption.\n");                                              |     826
  827     3 |  printf("Disable Link was called & the program was terminated.\n\n");                           |     827
  828       |                                                                                                 |     828
  829     4 |  data=sigdata();                                                                                |     829
  830     5 |  data->sigact->xhalt=0;                                                                         |     830
  831     6 |  data->sigact->xrtntosgnler=0;                                                                  |     831
  832     7 |  data->sigact->xresigprior=0;                                                                   |     832
  833     8 |  data->sigact->xresigouter=0;                                                                   |     833
  834       |                                                                                                 |     834
  835       |}  /*  End handler Subroutine  */                                                                |     835
                                               * * * * *   E N D   O F   S O U R C E   * * * * *
```

Figure 4-2 (Part 17 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                 S1016897        12/19/90 09:01:25     Page    18
                                                    * * * * *   I N C L U D E S   * * * * *
INCNO   Include Name                          Last change        Actual Include Name
    1   header                                90/12/19 08:49:31  UDCS_APPLS/QCSRC/HEADER
    2   typedefs                              90/12/19 08:49:31  UDCS_APPLS/QCSRC/TYPEDEFS
    3   stdio.h                               90/11/12 17:24:55  QCC/H/STDIO
    4   stddef.h                              90/11/12 17:24:54  QCC/H/STDDEF
    5   errno.h                               90/11/12 17:24:49  QCC/H/ERRNO
    6   signal.h                              90/11/12 17:24:53  QCC/H/SIGNAL
    7   ctype.h                               90/11/12 17:24:49  QCC/H/CTYPE
    8   stdarg.h                              90/11/12 17:24:54  QCC/H/STDARG
    9   stdlib.h                              90/11/12 17:24:56  QCC/H/STDLIB
   10   signal.h                              90/11/12 17:24:53  QCC/H/SIGNAL
   11   xxasio.h                              90/11/12 17:24:57  QCC/H/XXASIO
   12   xxcvt.h                               90/11/12 17:24:58  QCC/H/XXCVT
   13   string.h                              90/11/12 17:24:56  QCC/H/STRING
   14   ctype.h                               90/11/12 17:24:49  QCC/H/CTYPE
   15   hexconv                               90/12/19 08:49:27  UDCS_APPLS/QCSRC/HEXCONV
   16   stdio.h                               90/11/12 17:24:55  QCC/H/STDIO
                                          * * * * *   E N D   O F   I N C L U D E S   * * * * *
```

Figure 4-2 (Part 18 of 19). C/400 Compiler Listing for the Target Application

```
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                 S1016897       12/19/90 09:01:25     Page    19
                                         * * * * *   M E S S A G E   S U M M A R Y   * * * * *
     Total      Info(0-4)          Warning(5-19)       Error(20-29)        Severe(30-39)       Terminal(40-99)
         0          0                   0                   0                   0                   0
                                     * * * * *   E N D   O F   M E S S A G E   S U M M A R Y   * * * * *
5738CX1 V2R1M0  910329   IBM SAA C/400          UDCS_APPLS/TARGET                 S1016897       12/19/90 09:01:25     Page  20
 ROUTINE         BLOCK NUMBER  SCOPE  TYPE
 <MAIN>                 2      LOCAL  MAIN-PROGRAM
 __sigdata              6      LOCAL  PROCEDURE
 __sgnl                12      LOCAL  PROCEDURE
 __frdinit             49      LOCAL  PROCEDURE
 __getrec              50      LOCAL  PROCEDURE
 __putrec              51      LOCAL  PROCEDURE
 __frdexit             52      LOCAL  PROCEDURE
 __fwrinit             53      LOCAL  PROCEDURE
 __fwrexit             54      LOCAL  PROCEDURE
 QXXFORMAT            129      LOCAL  PROCEDURE
 __strlen             164      LOCAL  PROCEDURE
 __strncmp            168      LOCAL  PROCEDURE
 __strncpy            170      LOCAL  PROCEDURE
 inttohex             181      ENTRY   PROCEDURE
 hextoint             182      ENTRY   PROCEDURE
 sndformat1           197      ENTRY   PROCEDURE
 sndformat2           198      ENTRY   PROCEDURE
 setfilters           199      ENTRY   PROCEDURE
 byte                 200      ENTRY   PROCEDURE
 printespec           201      ENTRY   PROCEDURE
 settimer             202      ENTRY   PROCEDURE
 dequeue              203      ENTRY   PROCEDURE
 putdata              204      ENTRY   PROCEDURE
 x25lind              205      ENTRY   PROCEDURE
 disablelink          206      ENTRY   PROCEDURE
 handler              207      ENTRY   PROCEDURE
 main                 208      ENTRY   PROCEDURE
Program TARGET was created in library UDCS_APPLS.
                                   * * * * *   E N D   O F   C O M P I L A T I O N   * * * * *
```

Figure 4-2 (Part 19 of 19). C/400 Compiler Listing for the Target Application

The block numbers and explanations below correspond to those in the target application's program listing.

1 Call the C library routines fopen() and signal() to open the target file and set up a signal handler to process AS/400 exceptions, respectively. If an exception situation is encountered, the handler() will be called to perform clean-up in order for the program to end.

2 Call the QOLELINK API to enable the line description using the line name and communications handle passed as input parameters to this program.

3 Call the QOLTIMER API to time the completion of the enable link operation. If the timer expires before the enable-complete message is posted on the this program's data queue, then this program will end.

4 Call the QUSPTRUS API to obtain a pointer to the beginning of the output buffer user space. The output buffer will be used to construct a filter list for the call to the QOLSETF API.

5 Call the QOLRECV API to receive inbound data after reading an incoming data message that was posted on the program's data queue by the user-defined communications support. Since these programs are operating using the communications services of X.25, the first data unit the target program should see is a X'B201' operation signalling an incoming call was received.

6 Call the QOLSEND API with a X'B400' operation to accept the incoming X.25 call. A connection is now established between the source and target application programs.

7 The target program will now set a timer by calling the QOLTIMER API and wait for incoming data. If the timer expires before any incoming data is received, then this program will call the QOLDLINK API, and end.

8 This is the main receive loop for the target program. When data is received from the source program, it will be written to the target file opened during the initialization of this program. The loop will process until a message other than incoming-data entry is read from the program's data queue.

9 Call the QOLSEND API with a X'B001' operation to locally close the connection.

10 Receives a X'B101' operation from the user-defined communications support. This is a local confirmation of X'B100' operation.

11 Call the QOLDLINK API to disable the link previously enabled and end.

Chapter 5. Application Debugging

This section is intended to help an application programmer debug user-defined communications applications. It contains information on:

- System services and tools
- Error codes reported to the application program and QSYSOPR operation
- Common error list

System Services and Tools

There are several tools on the AS/400 system that may be useful for debugging the user-defined communications application. Some of the system provided tools which may be useful for developing user-defined communications applications are:

- Program Debug (STRDBG)
- Work with Job, Work with Communications Status (WRKJOB OPTION(*CMNSTS))
- Work with Job, Display Job Log (WRKJOB OPTION(*JOBLOG))
- Display Connection Status (DSPCNNSTS)
- Display Inbound Routing Data (using F6 after entering the DSPCNNSTS) command
- System Service Tools (STRSST)
  - Work with communications trace
  - Work with error log
- Dump System Object (DMPSYSOBJ)

Program Debug

Program debug (STRDBG) allows a programmer to trace the program and variables, set stops, change variables, and display variables. This function can be used to verify that the parameters are passed correctly.

Work with Communications Status

The Work with Job command, Work with Communications Status option, (WRKJOB OPTION(*CMNSTS)) shows the enabled links and operation counts for each link. Additionally, it reports information such as the communications handle the last operation requested, and the total input, output, and other operations requested. This information is shown for every link enabled by the job.

Display Job Log

The Work with Job command, selecting the Display job log option (WRKJOB OPTION(*JOBLOG)) is used to view the messages in the job log that will help determine the exact cause of the problem.

Display Connection Status

The Display Connection Status (DSPCNNSTS) will show information about the SVCs and PVCs that are in use by the application using the device description.

Note: The Display Line Description (DSPLIND) command can also show for each line, the switched virtual circuits that are in use, available, or attached to a controller description. This is not true for permanent virtual circuits.

Display Inbound Routing Data

Display Inbound Routing Data is accessed by using function key 6 (F6) while in the Display Connection Status (DSPCNNSTS) command. Using this command will show the results of the calls to QOLSETF. It can also help determine which device description has set a filter with duplicate inbound routing information.

Work with Communications Trace

The Work with Communications Trace function is part of the system service tools, accessed by issuing the Start System Service Tools (STRSST) command.

Work with Communications Trace will show data just as it appears to the network. If the application requests that data be sent and the request does not appear in the communications trace, the request will be rejected. The return and reason codes, and the error code reported in the parameter list of QOLSEND will indicate the reason the request was rejected.

Work with Error Log

The Work with Error Log function is part of the system service tools, accessed by issuing the Start System Service Tools (STRSST) command.

Some errors are reported by the system to the error log. A remote application that is communicating with a user-defined communications application on the local system, could cause an entry to be generated in the error log if one of the following conditions are met:

- When using a LAN, data is not received by the application and exceeds internal threshold values (3 Mb).
- When using an X.25 network, data is not received by the application and exceeds internal threshold values (128KB).

For both cases, the associated message in QSYSOPR will identify the error log that contains the error log entry. The error log entry contains information only.

Dump System Object to View User Spaces

The Dump System Object (DMPSYSOBJ) command can be used to inspect the user spaces after they are filled in by the application. The examples shown indicate what the user spaces look like for some of the operations.

User Space to Set a Filter to Route Inbound Data

This user space is filled in to activate two X.25 filters which will route any X.25 call containing X'BB', or X'DD' in the first byte of call user data (protocol ID byte).

```
5738SS1 V2R1M0  910524                     AS/400 DUMP          006625/QSECOFR/QPADEV0001     12/21/90 12:42:07      PAGE     1
DMPSYSOBJ PARAMETERS
OBJ- OUTBUFFER                    CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-           SPACE                                            *USRSPC
NAME-        OUTBUFFER                      TYPE-        19   SUBTYPE-         34
LIBRARY-     USRDFNCMN                      TYPE-        04   SUBTYPE-         01
CREATION-    12/21/90  12:40:03             SIZE-        00002200
OWNER-       QSECOFR                        TYPE-        08   SUBTYPE-         01
ATTRIBUTES-          0800                   ADDRESS-     00A00A00   0000
SPACE ATTRIBUTES-
  000000    00000080 00000060 1934D6E4 E3C2E4C6   C6C5D940 40404040 40404040 40404040  *       -  OUTBUFFER              *
  000020    40404040 40404040 E0000000 00000000   00002000 00800000 00000000 00000000  *        \                        *
  000040    00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *          (a                     *
SPACE-
  000000    01000002 001001BB 00000000 00000000   00000000 000001DD 00000000 00000000  *       Y                t        *
  000020    00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                 *
        LINES  000040    TO    001FFF  SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000    D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F5               *QPADEV0001QSECOFR   006625      *
SERVICE-
  000000    40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                      1          *
  000020    40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F44040  *          V2R1M00901221124004  *
  000040    40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                 *
  000060    40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                 *
  000080    00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                 *
  0000A0    00000000 00000000                                                          *                                 *
END OF DUMP
                                         * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 5-1. User Space to Set a Filter to Route Incoming X.25 Calls

User Space for X'B000' Operation, Initiating an SVC Call

The user space below has been filled in to initiate an SVC call specifying the following:

- default packet and window sizes
- D-bit (not selected)
- reverse charging (not selected)
- fast select (not selected)
- closed user group (not selected)
- other facilities (not selected)
- one byte of call user data, X'BB', which is the protocol identifier
- X.25 reset not supported by the user-defined communications application program
- 16KB is the maximum amount of contiguous data to be received
- automatic flow control value of 32

```
5738SS1 V2R1M0  910524                 AS/400 DUMP          006625/QSECOFR/QPADEV0001     12/21/90 12:47:42       PAGE
DMPSYSOBJ PARAMETERS
OBJ- OUTPUTBUF                     CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-          SPACE                                          *USRSPC
NAME-        OUTPUTBUF                    TYPE-         19   SUBTYPE-         34
LIBRARY-     USRDFNCMN                    TYPE-         04   SUBTYPE-         01
CREATION-    12/21/90  12:36:28           SIZE-         00001200
OWNER-       QSECOFR                      TYPE-         08   SUBTYPE-         01
ATTRIBUTES-          0800                 ADDRESS-      00A00100   0000
SPACE ATTRIBUTES-
  000000   00000080 00000060 1934D6E4 E3D7E4E3   C2E4C640 40404040 40404040 40404040  *        -  OUTPUTBUF            *
  000020   40404040 40404040 E0000000 00000000   00001000 00800000 00000000 00000000  *        \                       *
  000040   00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *           (a                   *
SPACE-
  000000   02000000 FFFFFFFF FFFFFFFF 00000000   00000008 40100001 00000000 00000000  *                                *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
       LINES  000040     TO     0000BF  SAME AS ABOVE
  0000C0   00000000 00000000 00000000 00000000   00000000 00000001 BB000000 00000000  *                         Y      *
  0000E0   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
       LINES  000100     TO     0001BF  SAME AS ABOVE
  0001C0   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00004000  *                                *
  0001E0   00200000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
  000200   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
       LINES  000220     TO     000FFF  SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000   D8D7C1C4 C5E5F0F0 F0F2D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F7               *QPADEV0002QSECOFR   006627      *
SERVICE-
  000000   40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                      1         *
  000020   40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F3F6 F2F84040  *           V2R1M00901221123628 *
  000040   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                *
  000060   40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                *
  000080   00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                *
  0000A0   00000000 00000000                                                          *                                *
END OF DUMP
                                          * * * * *  E N D   O F   L I S T I N G  * * * * *
```

Figure 5-2. User Space to Send an SVC Call

User Space Containing an Incoming X.25 Call, Operation X'B201'

This user space shows following:

- the call is using SVC 005
- both transmit and receive packet sizes are 128
- both transmit and receive window sizes are 7
- the calling DTE address is 40100000
- no other facilities are requested
- one byte of call user data, X'BB', which is the protocol identifier

The application received this call because it had set a filter to indicate to the system that it should route incoming X.25 calls that have the first byte of call user data (the protocol identifier) equal to X'BB' to the application.

```
5738SS1 V2R1M0  910524                      AS/400 DUMP          006625/QSECOFR/QPADEV0001     12/21/90 12:47:55      PAGE
DMPSYSOBJ PARAMETERS
OBJ- INBUFFER                        CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-          SPACE                                              *USRSPC
NAME-       INBUFFER                        TYPE-         19   SUBTYPE-        34
LIBRARY-    USRDFNCMN                       TYPE-         04   SUBTYPE-        01
CREATION-   12/21/90  12:40:03              SIZE-         00002200
OWNER-      QSECOFR                         TYPE-         08   SUBTYPE-        01
ATTRIBUTES-           0800                  ADDRESS-      00A00400   0000
SPACE ATTRIBUTES-
  000000   00000080 00000060 1934C9D5 C2E4C6C6   C5D94040 40404040 40404040 40404040  *       -  INBUFFER                *
  000020   40404040 40404040 E0000000 00000000   00002000 00800000 00000000 00000000  *       \                          *
  000040   00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *          (a                      *
SPACE-
  000000   00000005 00800007 00800007 00000000   00000008 40100000 00000000 00000000  *                                  *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                  *
       LINES  000040     TO     0000BF   SAME AS ABOVE
  0000C0   00000000 00000000 00000000 00000000   00000000 00000001 BB000000 00000000  *                        Y         *
  0000E0   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                  *
       LINES  000100     TO     001FFF   SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000  D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F5                *QPADEV0001QSECOFR   006625        *
SERVICE-
  000000  40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                       1          *
  000020  40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F34040  *          V2R1M00901221124003  *
  000040  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                  *
  000060  40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                  *
  000080  00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                  *
  0000A0  00000000 00000000                                                          *                                  *
END OF DUMP
                                         * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 5-3. User Space Containing an Incoming X.25 Call

User Space to accept an Incoming X.25 Call, Operation X'B400'

This user space was filled in to accept the incoming call, request default packet and window sizes, and no other additional facilities. The a maximum amount of contiguous data is set at 16KB and the automatic flow control is set at 32.

```
5738SS1 V2R1M0  910524                     AS/400 DUMP          006625/QSECOFR/QPADEV0001      12/21/90 12:48:06        PAGE
DMPSYSOBJ PARAMETERS
OBJ- OUTBUFFER                      CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-           SPACE                                         *USRSPC
NAME-       OUTBUFFER                     TYPE-         19   SUBTYPE-         34
LIBRARY-    USRDFNCMN                     TYPE-         04   SUBTYPE-         01
CREATION-   12/21/90  12:40:03            SIZE-         00002200
OWNER-      QSECOFR                       TYPE-         08   SUBTYPE-         01
ATTRIBUTES-         0800                  ADDRESS-      00A00A00  0000
SPACE ATTRIBUTES-
  000000   00000080 00000060 1934D6E4 E3C2E4C6   C6C5D940 40404040 40404040 40404040  *        -  OUTBUFFER               *
  000020   40404040 40404040 E0000000 00000000   00002000 00800000 00000000 00000000  *        \                          *
  000040   00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *           (a                       *
SPACE-
  000000   00000000 FFFFFFFF FFFFFFFF 00000000   00000000 00000000 00000000 00000000  *                                   *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                   *
        LINES  000040    TO    0001BF  SAME AS ABOVE
  0001C0   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00004000  *                                   *
  0001E0   00200000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                   *
  000200   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                   *
        LINES  000220    TO    001FFF  SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000   D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F5               *QPADEV0001QSECOFR   006625        *
SERVICE-
  000000   40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                      1            *
  000020   40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F44040  *           V2R1M00901221124004 *
  000040   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                   *
  000060   40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                   *
  000080   00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                   *
  0000A0   00000000 00000000                                                          *                                   *
END OF DUMP
                                          * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 5-4. User Space to Accept an Incoming X.25 Call

User Spaces for Sending Data, Operation X'0000'

Two user spaces are shown below. The first is the output buffer and the second is the output buffer descriptor.

The user spaces below are filled in to send three data units of 512 bytes each. The first two data units have the more data indicator turned on, indicating that all the data units are contiguous.

Note: This link was enabled, specifying a data unit size of 512 bytes.

```
5738SS1 V2R1M0  910524                    AS/400 DUMP          006625/QSECOFR/QPADEV0001     12/21/90 12:55:19      PAGE     1
DMPSYSOBJ PARAMETERS
OBJ- OUTPUTBUF                         CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-          SPACE                                          *USRSPC
NAME-        OUTPUTBUF                    TYPE-         19   SUBTYPE-          34
LIBRARY-     USRDFNCMN                    TYPE-         04   SUBTYPE-          01
CREATION-    12/21/90  12:36:28           SIZE-         00001200
OWNER-       QSECOFR                      TYPE-         08   SUBTYPE-          01
ATTRIBUTES-          0800                 ADDRESS-      00A00100   0000
SPACE ATTRIBUTES-
  000000   00000080 00000060 1934D6E4 E3D7E4E3   C2E4C640 40404040 40404040 40404040  *       -  OUTPUTBUF                *
  000020   40404040 40404040 E0000000 00000000   00001000 00800000 00000000 00000000  *        \                          *
  000040   00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *           (a                      *
SPACE-
  000000   F0F10000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *01                                *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                  *
        LINES  000040    TO     0001FF  SAME AS ABOVE
  000200   F0F20000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *02                                *
  000220   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                  *
        LINES  000240    TO     0003FF  SAME AS ABOVE
  000400   F0F30000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *03                                *
  000420   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                  *
        LINES  000440    TO     000FFF  SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000   D8D7C1C4 C5E5F0F0 F0F2D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F7               *QPADEV0002QSECOFR   006627        *
SERVICE-
  000000   40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                    1             *
  000020   40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F3F6 F2F84040  *           V2R1M00901221123628   *
  000040   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                  *
  000060   40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                  *
  000080   00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                  *
  0000A0   00000000 00000000                                                          *                                  *
END OF DUMP
                                            * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 5-5. User Space (Buffer) to Send Three Data Units

```
5738SS1 V2R1M0  910524                    AS/400 DUMP         006625/QSECOFR/QPADEV0001     12/21/90 12:55:58         PAGE
DMPSYSOBJ PARAMETERS
OBJ- OUTPUTBUFD                       CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-          SPACE                                        *USRSPC
NAME-        OUTPUTBUFD                   TYPE-         19   SUBTYPE-         34
LIBRARY-     USRDFNCMN                    TYPE-         04   SUBTYPE-         01
CREATION-    12/21/90  12:36:27           SIZE-         00000400
OWNER-       QSECOFR                      TYPE-         08   SUBTYPE-         01
ATTRIBUTES-          0800                 ADDRESS-      009FFE00  0000
SPACE ATTRIBUTES-
  000000   00000080 00000060 1934D6E4 E3D7E4E3   C2E4C6C4 40404040 40404040 40404040  *        -  OUTPUTBUFD               *
  000020   40404040 40404040 E0000000 00000000   00000200 00800000 00000000 00000000  *        \                           *
  000040   00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *           (a                       *
SPACE-
  000000   02000100 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                    *
  000020   02000100 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                    *
  000040   02000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                    *
  000060   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                    *
        LINES  000080    TO    0001FF  SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000   D8D7C1C4 C5E5F0F0 F0F2D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F7               *QPADEV0002QSECOFR   006627         *
SERVICE-
  000000   40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                   1               *
  000020   40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F3F6 F2F74040  *           V2R1M00901221123627    *
  000040   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                    *
  000060   40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                    *
  000080   00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                    *
  0000A0   00000000 00000000                                                           *                                    *
END OF DUMP
                                          * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 5-6. User Space (Descriptor Element) to Describe the Three Data Units

User Spaces for Receiving Data, Operation X'0001'

Two user spaces are shown below. The first is the input buffer and the second is the input buffer descriptor.

The user spaces below are filled in showing that 2 data units were received. The first data unit has the more data indicator turned on in the buffer descriptor for the data unit. This means that the X.25 more indicator was turned on in all the X.25 packets that this data unit contains. The second data unit does not have the more data indicator turned on, indicating that the last X.25 packet in the data unit had the X.25 more indicator turned off. The first and second data unit are considered to be logically contiguous to the user-defined communications application program.

Note: This link was enabled specifying a data unit size of 1024 bytes. The sending system sent the data in data unit sizes of 512 bytes and they were combined into the 1024 byte data unit size by the local system. The data unit size is not negotiated end-to-end, neither is the maximum amount of contiguous data or the automatic flow control. Because the values are important, each application should be aware of what the other application has specified for each value. Refer to "Sending and Receiving Data Packets" on page 3-12 for more information.

```
5738SS1 V2R1M0  910524                    AS/400 DUMP          006625/QSECOFR/QPADEV0001     12/21/90 12:59:33       PAGE    1
DMPSYSOBJ PARAMETERS
OBJ- INBUFFER                       CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-          SPACE                                               *USRSPC
NAME-        INBUFFER                      TYPE-          19   SUBTYPE-          34
LIBRARY-     USRDFNCMN                     TYPE-          04   SUBTYPE-          01
CREATION-    12/21/90  12:40:03            SIZE-          00002200
OWNER-       QSECOFR                       TYPE-          08   SUBTYPE-          01
ATTRIBUTES-         0800                   ADDRESS-       00A00400   0000
SPACE ATTRIBUTES-
  000000   00000080 00000060 1934C9D5 C2E4C6C6   C5D94040 40404040 40404040 40404040  *        -  INBUFFER              *
  000020   40404040 40404040 E0000000 00000000   00002000 00800000 00000000 00000000  *         \                       *
  000040   00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *           (a                    *
SPACE-
  000000   F0F10000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *01                              *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
        LINES  000040    TO    0001FF  SAME AS ABOVE
  000200   F0F20000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *02                              *
  000220   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
        LINES  000240    TO    0003FF  SAME AS ABOVE
  000400   F0F30000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *03                              *
  000420   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
        LINES  000440    TO    001FFF  SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000   D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F5               *QPADEV0001QSECOFR   006625      *
SERVICE-
  000000   40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                         1      *
  000020   40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F34040  *        V2R1M00901221124003     *
  000040   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                *
  000060   40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                *
  000080   00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                *
  0000A0   00000000 00000000                                                          *                                *
END OF DUMP
                                   * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 5-7. User Space (Buffer) Containing the Three Data Units

```
5738SS1 V2R1M0  910524                  AS/400 DUMP          006625/QSECOFR/QPADEV0001      12/21/90 12:59:41          PAGE
DMPSYSOBJ PARAMETERS
OBJ- INBUFFERD                      CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-           SPACE                                          *USRSPC
NAME-        INBUFFERD                   TYPE-         19  SUBTYPE-         34
LIBRARY-     USRDFNCMN                   TYPE-         04  SUBTYPE-         01
CREATION-    12/21/90  12:40:03          SIZE-         00000400
OWNER-       QSECOFR                     TYPE-         08  SUBTYPE-         01
ATTRIBUTES-          0800                ADDRESS-      00A00200  0000
SPACE ATTRIBUTES-
  000000   00000080 00000060 1934C9D5 C2E4C6C6   C5D9C440 40404040 40404040 40404040  *       -  INBUFFERD              *
  000020   40404040 40404040 E0000000 00000000   00000200 00800000 00000000 00000000  *        \                        *
  000040   00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *           (a                    *
SPACE-
  000000   04000100 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                 *
  000020   02000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                 *
  000040   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                 *
        LINES  000060    TO     0001FF  SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000   D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F5             *QPADEV0001QSECOFR   006625      *
SERVICE-
  000000   40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                    1            *
  000020   40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F34040  *           V2R1M00901221124003 *
  000040   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                 *
  000060   40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                 *
  000080   00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                 *
  0000A0   00000000 00000000                                                          *                                 *
END OF DUMP
                                    * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 5-8. User Space (Descriptor Element) Describing the Three Data Units

User Space to Clear a Connection or Call, Operation X'B100'

This user space was filled in to end an SVC connection or clear an incoming call. No facilities or clear user data are requested with this, but cause and diagnostic codes are specified (these are not ISO or SNA codes).

```
5738SS1 V2R1M0  910524                      AS/400 DUMP           006625/QSECOFR/QPADEV0001      12/21/90 13:14:48      PAGE    1
DMPSYSOBJ PARAMETERS
OBJ- OUTBUFFER                       CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-           SPACE                                           *USRSPC
NAME-        OUTBUFFER                       TYPE-          19   SUBTYPE-           34
LIBRARY-     USRDFNCMN                       TYPE-          04   SUBTYPE-           01
CREATION-    12/21/90  12:40:03              SIZE-          00002200
OWNER-       QSECOFR                         TYPE-          08   SUBTYPE-           01
ATTRIBUTES-          0800                    ADDRESS-       00A00A00    0000
SPACE ATTRIBUTES-
  000000   00000080 00000060 1934D6E4 E3C2E4C6   C6C5D940 40404040 40404040 40404040  *       -  OUTBUFFER             *
  000020   40404040 40404040 E0000000 00000000   00002000 00800000 00000000 00000000  *      \                         *
  000040   00000000 00000000 0005004D 42000400   00000000 00000000 00000000 00000000  *           (a                   *
SPACE-
  000000   0000BEBE 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *  XX                            *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
        LINES  000040    TO    001FFF  SAME AS ABOVE
POINTERS-
  NONE
OIR DATA-
TEXT-
  000000   D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6   D9404040 F0F0F6F6 F2F5               *QPADEV0001QSECOFR   006625      *
SERVICE-
  000000   40404040 40404040 40404040 40404040   40404040 40F14040 40404040 40404040  *                     1          *
  000020   40404040 40404040 404040E5 F2D9F1D4   F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F44040  *           V2R1M00901221124004  *
  000040   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                                *
  000060   40404040 40404040 40404040 40404040   40404040 40404040 00000000 00000000  *                                *
  000080   00000000 00000000 00000000 00000000   00000000 00000000 40400000 00000000  *                                *
  0000A0   00000000 00000000                                                          *                                *
END OF DUMP
                                          * * * * *  E N D   O F   L I S T I N G  * * * * *
```

Figure 5-9. User Space to Send an SVC Clear

Error Codes

The system and user-defined communications support will report important information which can be useful in determining recovery actions. These are typically referred to as error codes and are reported either to the job log or to the QSYSOPR message queue.

In some cases error codes are reported to the application in the error specific parameter. This section lists the valid error codes. Some of the error codes represent actual coding errors, others only report additional information.

LAN Error Codes

Table 5-1 on page 5-12 shows the hexadecimal codes which are valid for an application to receive as a result of a call to QOLSEND using operation code X'0000'. They indicate that the data was never sent on the line. Associated with these error codes is a message in QSYSOPR, indicating the device description that caused the error, and the error code. After receiving the error code, the link will still be enabled and usable.

To an application, these error codes indicate that a coding error was made and should be corrected.

Table 5-1. Error codes received while sending data over LAN

Error Code	Description	Cause
3300 2A55	Routing length not valid	Routing length is not valid, or length does not equal length in routing field.
3300 2A5D	Maximum frame size limit exceeded	Length of data is greater than maximum frame size supported by the source SAP
5300 2A7B	Access Control not valid	Access Control specified is not supported
3300 2AA9	SAP address not valid	SAP address is not configured in the line description
3300 2AA9	SAP address not valid	SAP address is not configured in the line description
3300 2AD4	Data length too small (Ethernet Version 2 only)	Data must be at least 48 bytes long (46 bytes of data, plus 2 bytes for the Ethernet type field)
3300 2AD5	Ethernet type field is not valid (Ethernet Version 2 only)	Ethernet type field (first two bytes of data)

X.25 Error Codes

Table 5-2 shows the error codes which are valid for an application to receive as a result of a call to QOLSEND with operation X'B400' to accept an SVC call, or a call to QOLRECV which returns the results of the open connection request operation, X'B101', or the connection failure indication, reported by operation X'B301'.

To an application, these error codes indicate that a coding error was made, or a failure condition occurred.

Table 5-2 (Page 1 of 2). Error codes reported on X'B001', X'B301', and X'B400' operations

Error Code	Description	Cause
3200 3050	Restart in progress	Temporary condition; retry operation
3200 3172	Outgoing channel not available	Temporary condition; retry operation
3200 3368	Remote address length not valid	Remote address length not supported by the network
3200 3384	Facility field error	A facility was encoded incorrectly or a duplicate facility was encoded
3200 3388	Facility field too long	The total length of the facilities, which includes user-specified facilities, the NUI facility from the line description, and system generated facilities, exceeded X.25 limits (109 bytes)
3200 338C	Response restricted by fast select	User data is not allowed with restriction

Table 5-2 (Page 2 of 2). Error codes reported on X'B001', X'B301', and X'B400' operations

Error Code	**Description**	**Cause**
3200 3394	User data not allowed	User data is not allowed on the call accept if fast select was not requested.
3200 33CC	Call user data length not valid	The length of call user data is greater than 16 and fast select is not selected.
4200 3210	Reset request transmitted	The virtual circuit was reset by the local system. Refer to cause and diagnostic codes to determine recovery.
4200 3220	Clear request transmitted	The virtual circuit was cleared by the local system. Refer to cause and diagnostic codes to determine recovery.
4200 3230	Restart request transmitted	The virtual circuit was cleared by the local system. Refer to cause and diagnostic codes for more information.
4200 3280	Time-out on call	Call timed out
4600 3134	Clear indication was received	The virtual circuit was cleared by either the remote system or the network. Refer to cause and diagnostic codes for more information.
4600 3138	Restart indication received	Temporary condition; refer to the cause and diagnostic codes reported to correct the problem, then retry the operation
BADD 4D10	The maximum data unit assembly size was exceeded by a remote system; the connection has been cleared (SVC) or reset (PVC).	A remote system has sent many X.25 packets to the local system with the more indicator on. The combined total length of the data in this sequence of packets, which have the more indicator on, has exceeded the maximum data unit assembly size specified by the user-defined communications application on the open connection request (X'B000').

Table 5-3 on page 5-14 shows the error codes that are valid for an application to receive as a result of a call to QOLRECV with an operation code returned as X'B101'.

To an application, these error codes indicate that the connection was cleared or reset for the following reasons.

Table 5-3. Error codes reported on the X'B101' operation

Error Code	Description	Cause
3200 3388	Facility field too long	The total length of the facilities, which includes user-specified facilities, the NUI facility from the line description, and system generated facilities, exceeded X.25 limits (109 bytes)
3200 3394	User data not allowed	User data is not allowed when fast select is not selected.
3200 33CC	Call user data length not valid	The length of call user data is greater than 16 and fast select is not selected.
4200 3240	Time-out on reset	The clear request resulted in an X.25 reset, which timed out
4200 3284	Time-out on clear	The remote system did not respond to the CLEAR within the time-out value
4600 3134	Clear indication was received	The virtual circuit was cleared by either the remote system or the network. Refer to cause and diagnostic codes for more information.

Table 5-4 shows the error codes that are valid for an application to receive as a result of a call to QOLRECV, returning the operation code, X'BF01'.

To an application, these error codes indicate that the connection was cleared or reset for the following reasons.

Table 5-4 (Page 1 of 2). Error codes reported on the X'BF01' operation

Error Code	Description	Cause
3200 3050	Network Restart in progress	Temporary condition; connection is no longer active.
3200 3A0C	Close pending	The virtual circuit is being closed.
3200 3A0D	Reset pending	The virtual circuit is in the process of being reset by either the remote system or the network.
4200 3210	Reset packet transmitted	A Reset packet was transmitted from the local system.
4200 3240	Time-out on reset	The clear request resulted in an X.25 reset, which timed out
4600 3130	Reset indication was received	The virtual circuit received a reset by either the remote system or the network. Refer to cause and diagnostic codes for more information.

Table 5-4 (Page 2 of 2). Error codes reported on the X'BF01' operation		
Error Code	**Description**	**Cause**
4600 3134	Clear indication was received	The virtual circuit was cleared by either the remote system or the network. Refer to cause and diagnostic codes for more information.

Table 5-5 shows the error codes that are valid for an application to receive as a result of a call to QOLSEND with an operation code returned as X'0000'.

To an application, these error codes indicate that the connection was cleared or reset for the following reasons.

Table 5-5. Error codes resulting from a X'0000' operation		
Error Code	**Description**	**Cause**
3200 3050	Network Restart in progress	Temporary condition; connection is no longer active.
3200 33C8	Data length not valid	The length of the packet is not supported for this virtual circuit.
3200 3A0C	Close pending	The virtual circuit is being closed.
3200 3A0D	Reset pending	The virtual circuit is in the process of being reset by either the remote system or the network.
4200 3284	Interrupt timed out	The local DTE sent an interrupt packet. The response to this packet was not received within the time-out period, and the connection has been reset by the AS/400 system.
4600 3130	Reset indication was received	The virtual circuit received a reset by either the remote system or the network. Refer to cause and diagnostic codes for more information.
4600 3134	Clear indication was received	The virtual circuit was cleared by either the remote system or the network. Refer to cause and diagnostic codes for more information.

Common Errors

This section shows some of the common errors that an application programmer may encounter. Some of these errors are detected by the APIs and reported to the application by the unsuccessful return and reason codes returned on each API. Other errors are program design errors, that the application programmer must detect and correct. The errors are listed by category:

Parameter errors

- Switching use of PCEP and UCEP
- Switching use of timer handles
- Not encoding parameters if not used
- Operation code not in hexadecimal format
- Parameter not declared with proper length

User Space errors

- Not encoding reserved space for fields not used
- Not initializing user space fields as necessary.

  The output user spaces can only be changed by the user-defined communications application. Operations are validated on each request. If there are fields that the current operation does not use, they should be set to contain zeros with X'00', to prevent a template error resulting from information on the previous operation still being in the user space. Not resetting the indicators in the output buffer descriptors on each operation and not zeroing out fields before making a call request may result in template errors.

Data Queue errors

- Data queue not created
- Data queue created with different key length than specified in the parameter list of QOLELINK

QOLRECV errors

- Not checking the more data output parameter and issuing another call to QOLRECV
- Not calling QOLSETF to set the filter to route inbound data to the application
- Using the wrong data unit descriptor for the data unit (each data unit has its own descriptor)

QOLSEND errors

- After a call to QOLSEND with operation codes of X'B000', X'B100', or X'BF00', the application should then call QRCVDTAQ and wait for an incoming data entry. The success or failure of these operations are reported through QOLRECV with operation codes of X'B001', X'B101' and X'BF01', respectively.
- Using the wrong data unit descriptor for the data unit (each data unit has its own descriptor)

QOLELINK errors

- User space names not unique
- Data queue not created before program call
- Line description not created or incorrect prior to program call

QOLQLIND errors

- Parameter buffer not large enough

Chapter 6. Configuration and Additional Information

This chapter describes how to configure user-defined communications support and the entries that user-defined communications support can send to the data queue.

Configuring User-Defined Communications Support

This section describes what needs to be configured before a user-defined communications application program can use the user-defined communications APIs. You can either use the system-supplied menus or the Control Language (CL) commands to do this configuration.

Link

A link allows a user-defined communications application program to use an X.25, token-ring, or Ethernet communications line and is made up of the following communications objects:

- an X.25, token-ring, or Ethernet line description
- a network controller description
- a network device description of type *USRDFN

It is only necessary to configure the line description since user-defined communications support will automatically configure a network controller and network device description of type *USRDFN when the link is enabled. Additionally, the line, controller, and device description will be automatically varied on if necessary.

The following commands are used to create or change line descriptions:

- CRTLINX25 — Create Line Description (X.25)
- CHGLINX25 — Change Line Description (X.25)
- CRTLINTRN — Create Line Description (token-ring)
- CHGLINTRN — Change Line Description (token-ring)
- CRTLINETH — Create Line Description (Ethernet)
- CHGLINETH — Change Line Description (Ethernet)

The following commands may be used to create or change controller descriptions:

- CRTCTLNET — Create Controller Description (Network)
- CHGCTLNET — Change Controller Description (Network)

The following commands may be used to create or change device descriptions:

- CRTDEVNET — Create Device Description (Network)
- CHGDEVNET — Change Device Description (Network)

See the *OS/400* Communications Configuration Reference* for more information on configuring communications.

Data Queue

A data queue is used by user-defined communications support to inform a user-defined communications application program of some action to take or of an activity that has been completed, and must be created before the link is enabled.

The size of each data queue entry must be large enough to accommodate the user-defined communications support entries. Refer to "Data Queue Entries" for more information on the entries that user-defined communications support can send to the data queue.

The Create Data Queue (CRTDTAQ) command is used to create data queues. See the *CL Reference* for more information on the CRTDTAQ command.

Data Queue Entries

This section describes the entries user-defined communications support can send to the data queue.

General Format

User-defined communications support informs a user-defined communications application program of some action to take or of an activity that has been completed by sending an entry to the data queue.

The length of each user-defined communications support entry will always be at least 80 bytes. When using a keyed data queue, however, each entry may be as large as 336 bytes, depending on the size of the key value supplied to the user-defined communications support.

Figure 6-1 shows the general format of each user-defined communications support entry.

	Entry type CHAR(10)	Entry ID CHAR(2)	Entry data CHAR(68)	Key CHAR(256)
Bytes	1–10	11–12	13–80	81–336

Figure 6-1. Data Queue Entry General Format

Entry type: This indicates the type of entry on the data queue and will be *USRDFN for all user-defined communications support entries.

Entry ID: This uniquely identifies each entry within an entry type. User-defined communications support has five entries defined:

- enable-complete entry (entry ID = '00')
- disable-complete entry (entry ID = '01')
- permanent-link-failure entry (entry ID = '02')
- incoming-data entry (entry ID = '03')
- timer-expired entry (entry ID = '04')

Note: The entry type of *USRDFN and all associated entry IDs, either defined or undefined, are reserved for the user-defined communications support. Therefore, the user-defined communications application program should not define entries using this entry type.

Entry data: This data is useful to the user-defined communications application program and varies according to the entry ID.

Key: When using a keyed data queue, this is the key value supplied to the user-defined communications support.

Enable-Complete Entry

The enable-complete entry is sent to the data queue when the enable link operation has completed. This entry will only be sent after the QOLELINK program returns to the user-defined communications application program with a successful return and reason code.

Note: The QOLELINK program only initiates the enabling of the link. The user-defined communications application program must wait for the enable-complete entry before attempting to perform input or output on the link.

Figure 6-2 shows the format of the enable-complete entry.

*USRDFN	'00'	Communications handle	Status	Resrvd	Key
Bytes 1-10	11-12	13-22	23	24-80	81-336

Figure 6-2. Enable-Complete Entry

Communications handle: The name of the link that is being enabled. This was supplied by the user-defined communications application program when the QOLELINK program was called.

Status: This indicates the outcome of the enable link operation. A value of zero indicates the enable link operation was successful and I/O is now possible on this link. A value of one indicates the enable link operation was not successful (the job log will contain messages indicating the reason). The user-defined communications support will disable the link when the enable link operation completes unsuccessfully and the disable-complete entry will not be sent to the data queue.

Key: The key value associated with the enable-complete entry when using a keyed data queue. This was supplied by the user-defined communications application program when the QOLELINK program was called. When using a non-keyed data queue, indicated by supplying a key length of zero to the QOLELINK program, this field would not be present.

Disable-Complete Entry

The disable-complete entry is sent to the data queue when a link has been successfully disabled. This entry will always be the last entry sent by the user-defined communications support on this link and, therefore, provides a way for the user-defined communications application program to remove any enable-complete, incoming-data, or permanent-link-failure entries previously sent to the data queue[1].

[1] User-defined communications support does not associate timers with links. Therefore, it is possible for a timer-expired entry to be sent to the data queue after the link is disabled. The user-defined communications application program is responsible for handling this.

Figure 6-3 shows the format of the disable-complete entry.

*USRDFN	'01'	Communications handle	Reserved	Key
Bytes 1–10	11–12	13–22	23–80	81–336

Figure 6-3. Disable-Complete Entry

Communications handle: The name of the link that has been disabled. This was supplied by the user-defined communications application program when the QOLELINK program was called to enable the link.

Key: The key value associated with the disable-complete entry, when using a keyed data queue. This was supplied by the user-defined communications application program when the QOLELINK program was called to enable the link. When using a nonkeyed data queue, indicated by supplying a key length of zero to the QOLELINK program, this field would not be present.

Permanent-Link-Failure Entry

The permanent-link-failure entry is sent to the data queue when error recovery has been canceled on a link. You must disable and enable the link to recover.

Figure 6-4 shows the format of the permanent-link-failure entry.

*USRDFN	'02'	Communications handle	Reserved	Key
Bytes 1–10	11–12	13–22	23–80	81–336

Figure 6-4. Permanent-Link-Failure Entry

Communications handle: The name of the link on which the failure has occurred. This was supplied by the user-defined communications application program when the QOLELINK program was called to enable the link.

Key: The key value associated with the permanent-link-failure entry, when using a keyed data queue. This was supplied by the user-defined communications application program when the QOLELINK program was called to enable the link. When using a nonkeyed data queue, indicated by supplying a key length of zero to the QOLELINK program, this field would not be present.

Incoming-Data Entry

The incoming-data entry is sent to the data queue when the user-defined communications support has data for the user-defined communications application program to receive. When this entry is received, the user-defined communications application program should call the QOLRECV program to pick up the data.

Note: Another incoming-data entry will not be sent to the data queue until the user-defined communications application program picks up all the data from the user-defined communications support. This is indicated by the data available parameter on the call to the QOLRECV program.

Figure 6-5 on page 6-5 shows the format of the incoming-data entry.

	*USRDFN	'03'	Communications handle	Reserved	Key
Bytes	1-10	11-12	13-22	23-80	81-336

Figure 6-5. Incoming-Data Entry

Communications handle: The name of the link on which the data has come in. This was supplied by the user-defined communications application program when the QOLELINK program was called to enable the link.

Key: The key value associated with the incoming-data entry, when using a keyed data queue. This was supplied by the user-defined communications application program when the QOLELINK program was called to enable the link. When using a nonkeyed data queue, indicated by supplying a key length of zero to the QOLELINK program, this field would not be present.

Timer-Expired Entry

The timer-expired entry is sent to the data queue when a timer, previously set by a user-defined communications application program, expires.

Figure 6-6 shows the format of the timer-expired entry.

	*USRDFN	'04'	Timer handle	User data	Key
Bytes	1-10	11-12	13-20	21-80	81-336

Figure 6-6. Timer-Expired Entry

Timer handle: The name of the expired timer. This was returned to the user-defined communications application program when the QOLTIMER program was called to set the timer.

User data: The data associated with the expired timer. This data was supplied by the user-defined communications application program when the QOLTIMER program was called to set the timer.

Key: The key value associated with the timer-expired entry, when using a keyed data queue. This was supplied by the user-defined communications application program when the QOLTIMER program was called to set the timer. When using a nonkeyed data queue, indicated by supplying a key length of zero to the QOLTIMER program, this field would not be present.

Part 2. Virtual Terminal Application Programming Interfaces

Chapter 7. Introduction to Virtual Terminal APIs

The purpose of this chapter is to provide information for using the virtual terminal (VT) APIs. The VT APIs allow an AS/400 user-written program to interact with an AS/400 application program which is performing work station input and output (I/O). This interaction is performed using a virtual terminal.

A virtual terminal is a device that does not have hardware associated with it. It is used to form a connection between a user-written program representing a physical work station (possibly on a remote system) and AS/400 applications. The virtual terminal is managed by the OS/400. Work station I/O performed by an AS/400 application is directed to the virtual terminal. The VT APIs allow another AS/400 application (called a server program) to work with the data associated with the virtual terminal.

The server program generally runs on behalf of (or in conjunction with) another program called a client program (either on the same AS/400 system or on some other remote system). In general, the server program and client program allow a work station to be supported as if the work station were connected locally.

The client and server programs may reside on the same AS/400 system or may be distributed between two different systems. Figure 7-1 shows a model for such a distributed implementation between a PS/2* work station and an AS/400 system. This implementation is similar to how PC Support Work Station Function is supported for the AS/400 system.

Implementation of Distributed 5250 Emulation Model

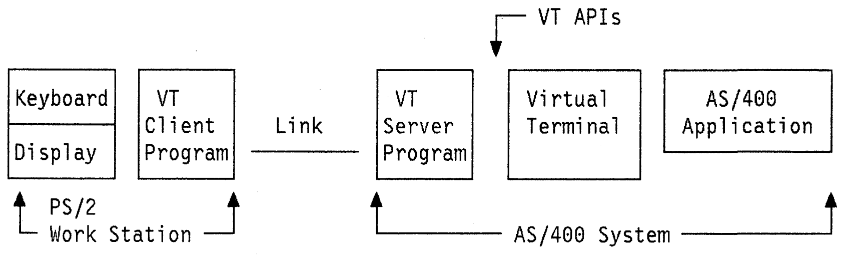


Figure 7-1. Example Virtual Terminal Client/Server Model

The work station in the model in Figure 7-1 is a personal computer such as the IBM PS/2.

The client program on this PS/2 work station does the following:

- accepts data from the server program and displays the data on the PS/2 display
- accepts data from the PS/2 keyboard and sends the data to the server program
- converts the data from the format required by the PS/2 display and keyboard to the format required by the server program (5250 data stream)

The link between the client program and server program uses DOS and OS/400 communications support. This may be LU6.2, TCP/IP, or some other communications protocol.

The server program provides the work station support of the server implementation. The server program runs on an AS/400 system and can be written in any AS/400 high-level language (HLL) such as C/400. The server program writes data to the virtual terminal and reads data from the virtual terminal using the VT APIs. The virtual terminal data is always in a 5250 data stream format.

The VT APIs provide an interface between the server program and the virtual terminal supported by OS/400.

The virtual terminal represents the OS/400 link between the server program and the AS/400 application. The virtual terminal is managed entirely by the OS/400.

The AS/400 application is an OS/400 or user-written application that performs a standard data transfer to an OS/400 work station. This application may be written in any OS/400 HLL.

Table 7-1 lists the available VT APIs. Refer to Chapter 10, "Virtual Terminal APIs" for additional information concerning these APIs.

Table 7-1. VT API Functions

API Name	**Description**
QTVOPNVT	Open Virtual Terminal Path. Opens a path to a virtual terminal, allowing a server program to interact with an AS/400 application.
QTVRDVT	Read Virtual Terminal Data. Reads data from the virtual terminal to the server program's data buffer.
QTVWRTVT	Write Virtual Terminal Data Writes data from the server program's data buffer to the virtual terminal.
QTVSNDRQ	Send Virtual Terminal Request to OS/400. Sends a request to OS/400 for a particular function to be performed.
QTVCLOVT	Close Virtual Terminal Device. Closes one or all open paths to virtual terminals.

Chapter 8. Getting ready for using the VT APIs

To get ready for using the VT APIs, perform the following steps:

1. Set the number of Automatically Created Virtual Terminals (QAUTOVRT)
2. Set the Limit Security Officer (QLMTSECOFR) system value
3. Create User Profiles

This chapter also discusses how to create your own virtual controllers and devices, and programming considerations when using the VT APIs.

Step 1 - Setting the Number of Automatically Created Virtual Terminals

Virtual terminals are used by the OS/400 to allow the server program to interact by sending and receiving data with AS/400 applications. The OS/400 will automatically select (and create if necessary) these virtual terminals for you.

You must allow OS/400 to configure the required virtual controllers and terminals automatically. Controllers coordinate and control the operation of one or more input/output terminals (such as work stations) and synchronize the operation of such terminals with the operation of the system as a whole. The QAUTOVRT system value specifies the maximum number of terminals that will be automatically configured by the system. Use the Change System Value (CHGSYSVAL) command to change the value of the QAUTOVRT system value. For example, entering the following command string changes the number of virtual terminals that can be allocated on a system to 50:

```
CHGSYSVAL SYSVAL(QAUTOVRT) VALUE(50)
```

To determine and set the maximum number of users you want signed on to the AS/400 system at any time do the following:

- Set the QAUTOVRT value to 9999, the maximum value allowed.
- Let your clients use the AS/400 system until you decide that the number of virtual terminals created is sufficient for normal system operation.
- Use the Work With Configuration Status (WRKCFGSTS) to determine the number of work stations configured.
- Change the QAUTOVRT value from 9999 to the number of virtual terminals you require for normal operation.

If you have never allowed virtual terminals to be configured automatically on your system, the QAUTOVRT value is 0. You will not be able to use the VT APIs because OS/400 will not be able to create more than the specified QAUTOVRT work stations (0). If you change the QAUTOVRT value to 10, the next virtual terminal path opened causes the OS/400 to create a virtual terminal. This virtual terminal is created because the number of virtual terminals on the controller (0) is less than the number of specified in the QAUTOVRT value (10). Even if you change the specified number to 0 again, the next virtual terminal opened may succeed if a virtual terminal exists that is not being used. If a virtual terminal does not exist or is in use, OS/400 will not create a new virtual terminal because the number of virtual terminals currently existing is greater than or equal to the specified QAUTOVRT value. If the number of virtual terminals currently existing is greater than or equal to the QAUTOVRT value, the message CPF8940, "Cannot automatically select virtual device", will be sent to the system operator message queue. You must either try again when a virtual description becomes available or increase the QAUTOVRT value.

The OS/400 uses the following conventions for naming virtual controllers and work stations:

- Virtual controllers are named QPACTLnn.
- Virtual terminal descriptions are named QPADEVxxxx.

The following must be considered when the OS/400 automatically configures work stations.

- The OS/400 does not delete virtual terminals, even if the number of work stations attached to virtual controllers that are automatically configured, exceeds the QAUTOVRT limit.

  If you want the extra work stations deleted, you must manually delete them.

- OS/400 allows a maximum of 250 virtual terminals on the QPACTL01 before it creates QPACTL02. This value is usually adequate for most users. If you delete work stations to enforce a smaller QAUTOVRT value, begin by deleting the work stations from the controller with the highest QPACTLnn value.

  Note: QAUTOVRT is a system value that controls the number of all automatically configured virtual terminals for the AS/400 system. Virtual terminals are used by TELNET, 5250 display station pass-through, and all other programs using the VT APIs.

Security Considerations

The number of sign-on attempts allowed increases if virtual terminals are automatically configured. The number of sign-on attempts is equal to the number of system sign-on attempts allowed, multiplied by the number of virtual terminals that can be created. The number of system sign-on attempts allowed is defined by the QMAXSIGN system value. The number of virtual terminals that can be created is defined by the QAUTOVRT system value.

Step 2 - Setting the Limit Security Officer (QLMTSECOFR) Value

The OS/400 supports the Limit Security Officer (QLMTSECOFR) system value, which limits the devices the security officer can sign on to. The security officer is assigned to control all of the security authorizations provided with the AS/400 system. If the QLMTSECOFR value is greater than zero, the security officer must be authorized to use the virtual device descriptions. However, when this value is 0, the system does not limit the devices the security officer can use to sign on to.

On AS/400 systems with a QSECURITY value of 30, a user with Security Officer Authority (*ALLOBJ) must be authorized to use work stations before the system allows the user to use those work stations. For example, for each display work station that a security officer wants to sign on to (local, remote, or virtual), the user must specify the following with the Grant Object Authority (GRTOBJAUT) command:

```
GRTOBJAUT OBJ(display-name) OBJTYPE(*DEVD) AUT(*CHANGE) USER(QSECOFR)
```

This procedure is very important because using the VT APIs automatically configures virtual terminals (devices). If the QLMTSECOFR value is set to 0, all virtual terminals automatically configured when using the VT APIs can be used by the security officer. If you set the QLMTSECOFR value to 1, your security officer will not be able to use the virtual terminals unless you grant object authority to the security officer for that virtual terminal. Note that the automatic configuration support can delete and recreate the virtual terminal. If this occurs, authority must be granted to the security officer each time the virtual terminal is created. Automatic configuration is a function that names and creates the descriptions of network devices and controllers attached to a line.

Step 3 - Creating User Profiles

On the AS/400 system, you should create one or more user profiles for users of the virtual terminal supported by the client and server programs. The default user profile is *SYS. The following example shows a sample user profile:

```
CRTUSRPRF  USRPRF(CLERK1)  PASSWORD(unique-password)
           JOBD(CLERKLIB/CLERKL1)
           TEXT('User profile for one group of clerks')
```

Creating Your Own Virtual Controllers and Devices

You can create your own virtual controllers and devices (terminals); however, you must use the same naming conventions as the automatic controller and device creation support. You may want to create the virtual terminal descriptions to control the number of sign-on attempts possible by not allowing automatic configuration of virtual terminals (which allows additional sign-on attempts to occur). Refer to "Security Considerations" on page 8-2 for additional information.

If you do not want to use automatically created descriptions, do the following:

- To create your own descriptions
  - To create a controller description for a virtual terminal, use the Create Controller Description for Virtual Work Station (CRTCTLVWS) command

    ```
    CRTCTLVWS CTLD(QPACTL01) TEXT('Virtual Controller for virtual terminals')
    ```

    Note: You must use the OS/400 naming convention, QPACTLnn, for naming virtual controllers, where nn is a decimal number starting at 01.
  - To create a virtual terminal, use the Create Terminal Display (CRTDEVDSP) command as follows:

    ```
    CRTDEVDSP DEVD(QPADEV0001) DEVCLS(*VRT) TYPE(5251) MODEL(11)
      CTL(QPACTL01) TEXT('24 X 80 Monochrome Display for Server Program')
    ```
  - OS/400 will automatically vary on the controller and terminal that you have created. You must use the OS/400 naming convention, QPADEVnnnn, for naming virtual device descriptions.
- After creating the descriptions, you must authorize the server program to use them. Use the Grant Object Authority (GRTOBJAUT) command to authorize the user profile used by the server program to the descriptions created earlier. This can be done with the following commands:

  ```
  GRTOBJAUT OBJ(QPACTL01) OBJTYPE(*CTLD) AUT(*CHANGE) USER(user_profile)

  GRTOBJAUT OBJ(QPADEV0001) OBJTYPE(*DEVD) AUT(*CHANGE) USER(user_profile)
  ```

- You may want to prevent virtual terminals from being created automatically. To do this, set the QAUTOVRT system value to 0 as follows:

  ```
  CHGSYSVAL SYSVAL(QAUTOVRT) VALUE(0)
  ```

  Refer to "Step 1 - Setting the Number of Automatically Created Virtual Terminals" on page 8-1 for additional information.

  Note: Changing this system value will affect other AS/400 products and programs requiring automatic configuration. This includes TELNET, 5250 display station pass-through, and other programs using the VT APIs.

Developing Server and Client Programs

The following should be considered when developing client and server programs:

- The client program should be able to:
  - Interrupt the server program
  - Check the server program's status
  - Discard data from the AS/400 application
- The user should be able to configure a time-out to be used by the client program while waiting for screens from the server program.
- Pressing the Print key on the work station should create a file to be printed at either the work station or the AS/400 printer.

Chapter 9. Before Using VT APIs

This chapter provides information that you will need when developing AS/400 programs that use the VT APIs. Chapter 10, "Virtual Terminal APIs," contains the syntax of the VT APIs called QTVOPNVT, QTVRDVT, QTVWRTVT, QTVSNDRQ, and QTVCLOVT.

Work Station Types

A server program can select from several different types of work stations. Refer to "Supported Work Station Types and Models" on page 10-3 for a list of the types of the work stations supported. A work station type must be specified when a virtual terminal device is opened.

Virtual Terminal Data

The VT APIs allow a server program to read and write virtual terminal data. This data is in a 5250 data stream format. Refer to the *5250 Functions Reference Manual*, SA21-9247, for more information on 5250 data streams.

Data Queues

Data queues are used by the OS/400 to send data to the server program. The data queue can also be used by the server program for interprocess communications with other AS/400 programs and APIs.

The data queue must be created by the user of VT APIs before a path is opened to a virtual terminal. Refer to the *CL Reference* for details on how to create and delete data queues. Refer to QTVOPNVT VT API for information on how a server program specifies the name of the data queue to be used.

The OS/400 communicates special events to the server program using the data queue. The information is provided by the AS/400 system using a data queue entry.

The following events result in data queue entries being sent.

- When data from an AS/400 application is received at a virtual terminal.
- The virtual terminal is being closed by the OS/400. This can occur if an AS/400 application's job is canceled.

When using VT APIs, data queues are used as follows:

1. The data queue is created by the user.
2. A path to a virtual terminal is opened by calling QTVOPNVT.
3. The server program should attempt to remove an entry from a queue by calling QRCVDTAQ (Receive Data Queue). Refer to the *CL Programmer's Guide* for information on how to call QRCVDTAQ.
4. If an entry is received indicating that data is available, the server program should read the data from the virtual terminal by calling QTVRDVT.

Table 9-1 shows the structure of OS/400 data queue entries sent to the data queue when using the VT APIs. All data queue entries have the same format.

Table 9-1. Format of OS/400 data queues associated with the VT APIs.

Name	**Type**	**Description**
Entry_Type	CHAR(10)	Always set by OS/400 to *VRTTRM.
Entry_ID	CHAR(2)	Entry ID associated with entry. Valid values for the 2 characters in this parameter are: 1. The OS/400 is closing (terminating) the session with the virtual terminal. The server program should perform a close to indicate that the server program is done using the virtual terminal. 2. An AS/400 application has sent data to the virtual terminal. Refer to the QTVRDVT API for information on how to read the data. **Note:** All other values are reserved by AS/400.
VTHandle	CHAR(16)	The virtual terminal handle associated with the virtual terminal communications path previously opened by calling QTVOPNVT. Refer to the QTVOPNVT VT for additional details.
	CHAR(52)	Reserved
Key	CHAR(*)	Key value specified when the virtual terminal communications path was opened, refer to QTVOPNVT API for additional details on specifying the key value. The key value can be used for retrieving data queues by key.

AS/400 Job Information

Several AS/400 jobs are involved when using VT APIs. The jobs can be classified into two groups: the server program jobs and the application jobs. Each group may consist of one or more jobs, depending on the implementation of the server program and the AS/400 applications being run.

The server program may be either implemented as a single job or more than one job, depending on the number of work stations to be supported per job. For example, each work station can be supported by a single job by routing all requests from the work station client program to a particular server program job.

The AS/400 application runs in its own job but there may be more than one application being run and, therefore, more than one job.

AS/400 Subsystem Information

The server program should be run in the same subsystem that other server programs are running in. A separate subsystem is usually best for all server programs for controlling resource use, such as main storage, and for allowing performance tuning, such as the number of page faults. Generally, AS/400 applications run in subsystem QBASE.

You may need to use the Add a Work Station Entry (ADDWSE) command to add a work station entry to the subsystem description under which you want the server program to run. Note that if the subsystem was created with the system defaults, this will not have to be done.

Before a work station is allowed to sign-on, it must be defined to the subsystem. The work station in this case is the virtual terminal device (QPADEVnnnn) automatically created by OS/400. The work station name, work station type, or *ALL must be specified in the subsystem description. Use the Display Subsystem Description (DSPSBSD) command to see the list of work station entries defined to a subsystem. The following command could be used to add all work station types to a subsystem named QBASE:

```
ADDWSE SBSD(QBASE) WRKSTNTYPE(*ALL)
```

The ADDWSE command is only valid when the subsystem description is not active.

Chapter 10. Virtual Terminal APIs

This chapter explains the details of using the following VT APIs:

QTVOPNVT	Open Virtual Terminal Path
QTVRDVT	Read from Virtual Terminal
QTVWRTVT	Write to Virtual Terminal
QTVSNDRQ	Send Request for OS/400 Function
QTVCLOVT	Close Virtual Terminal Path

Open Virtual Terminal Path API (QTVOPNVT)

The Open Virtual Terminal Path API (QTVOPNVT) opens a path to a virtual terminal, allowing a server program to interact with an OS/400 application. The virtual terminal path remains open until it is explicitly closed or the job is ended.

When you call QTVOPNVT, the operating system selects or automatically configures a virtual terminal for you and indicates that the device is logically turned on. The operating system then creates a sign-on display at the virtual terminal and sends a message to the specified data queue to signal the server program that data is available.

The QTVOPNVT API has these attributes:

Library Authority	*USE	
User Queue Authority	*CHANGE	
User Queue Lock	*EXCLRD	
Error Messages		
	CPF87FA E	Character identifier not valid
	CPF87F0 E	Virtual terminal type value &1 not valid
	CPF87F1 E	Queue key length &1 not valid
	CPF87F2 E	Virtual terminal handle &1 not valid
	CPF87F7 E	Parameter value &1 not valid
	CPF87F8 E	Unexpected internal system error occurred in program &1
	CPF87F9 E	Keyboard language type &1 not valid

The QTVOPNVT API has these parameters:

QTVOPNVT: Required Parameters			
1	**Virtual terminal handle**	**OUTPUT**	**CHAR(16)**
	A reference code created by the operating system to identify this open virtual terminal path in later calls to other virtual terminal APIs.		
2	**Keyboard language type**	**INPUT**	**CHAR(3)**
	The keyboard language type for the virtual terminal. To use the system value, specify blanks for this parameter. For a list of other valid values, see the Create Device Description (CRTDEVDSP) command in the *CL Reference*. For details about supported keyboard languages, see the *National Language Support Planning Guide*.		

3	**Character set**	**INPUT**	**BINARY(4)**
	The graphic character set for the virtual terminal. Valid values are a specific graphic character set number and these special values: **0** The character set system value is used. **-1** The keyboard language type is used to select the appropriate character set. For details about the graphic character sets you can specify, see the *National Language Support Planning Guide*.		
4	**Code page**	**INPUT**	**BINARY(4)**
	The code page for the virtual terminal. For details about the code pages you can specify, see the *National Language Support Planning Guide*. If parameter 3 is 0 or -1, you do not have to specify parameter 4. When you use 0 for parameter 3, the system value is used for parameter 4. When you use -1 for parameter 3, the code page value is derived from parameter 2 (keyboard language type). **Note:** The code-page system value is obtained from the QCVRID system value.		
5	**Work station type**	**INPUT**	**BINARY(4)**
	The type of work station to use. Valid values are 1 through 10; refer to "Supported Work Station Types and Models" on page 10-3 for an explanation of the values. Other work station types and models are supported. You can specify these by determining their equivalents in the list above. For a more detailed list of work station types and equivalents, see "Supported Work Station Types and Models" on page 10-3. If a virtual terminal description does not yet exist for the work station type specified, the operating system tries to configure the work station automatically. Automatic configuration is controlled by the QAUTOVRT system value, which specifies the number of virtual terminals that the operating system can configure automatically. To change the QAUTOVRT value, use the Change System Value (CHGSYSVAL) command. **Note:** 5250 display station pass-through and TCP/IP TELNET use the QAUTOVRT system value.		
6	**Data queue name and library**	**INPUT**	**CHAR(20)**
	The name and library of the data queue used by the application program to receive data from the operating system asynchronously. The first 10 bytes give the data queue name, and the second 10 bytes give the library name. Allowable special values are: ***CURLIB** The job's current library ***LIBL** The library list		
7	**Key value**	**INPUT**	**CHAR(*)**
	The key value to use for the OS/400 data queue.		
8	**Key value length**	**INPUT**	**BINARY(4)**
	The length of the key value. Valid values are 0 through 256. If you specify 0, no key is used for data queue entries.		
9	**Error code**	**I/O**	**CHAR(*)**
	The structure in which to return error information. For the format of the structure, see "Error Code Parameter" on page 10-11.		

Supported Work Station Types and Models

This table details the values you can specify for the QTVOPNVT API's work station type parameter:

Value	Work Station Type and Model	Equivalent Type and Model	Description
1	5251 11		24 x 80 monochrome display.
2	5291 1	5291 2	24 x 80 monochrome display.
3	5292 2		24 x 80 color graphics display. This type is also emulated by a graphics work station feature.
4	5555 B01	5555 E01	24 x 80 monochrome double-byte character set (DBCS) display. This type is emulated by a monochrome work station feature that supports a DBCS display.
5	3196 A1	3196 A2 3196 B1 3196 B2 3476 EA	24 x 80 monochrome display. This type is emulated by a monochrome work station feature. This is what the ASCI devices emulate.
6	3179 2	3197 C1 3197 C2 3476 EC 5292 1	24 x 80 color display. This type is emulated by a color work station feature.
7	3180 2	3197 D1 3197 D2 3197 W1 3197 W2	27 x 132 monochrome display.
8	3477 FC		27 x 132 wide-screen color display.
9	3477 FG	3477 FA 3477 FD 3477 FW 3477 FE	27 x 132 wide-screen monochrome display.
10	5555 C01	5555 F01	24 x 80 color double-byte character set (DBCS) display. This type is emulated by a color work station feature that supports a DBCS display.

Note:

All 5250 work stations, except 5555 B01, can operate as 5251 11 work stations.

Selecting double-byte character set (DBCS) work stations requires the AS/400 primary language to be one of the DBCS national language versions (NLVs).

Read from Virtual Terminal API (QTVRDVT)

The Read from Virtual Terminal API, QTVRDVT, reads data from the virtual terminal into the server program's data buffer. Your application should read data only if it has received an asynchronous notification message on the data queue, or if the more data flag was set on a previous read operation. The data received is in 5250 data stream format.

Only one full-screen display of data can be received at a time. If the data buffer is too small, partial displays are received and the more data flag of the QTVRDVT API's read information parameter is set to 1.

Before working with 5250 data streams, be sure to see the *IBM 5250 Information Display System Functions Reference Manual.*

The following error messages are issued by the QTVRDVT API:

CPF87F2 E	Virtual terminal handle &1 not valid
CPF87F3 E	Data buffer length &1 not valid
CPF87F7 E	Parameter value &1 not valid
CPF87F8 E	Unexpected internal system error occurred in program &1

The QTVRDVT API has these parameters:

QTVRDVT: Required Parameters			
1	**Virtual terminal handle**	**INPUT**	**CHAR(16)**
	The reference code for the open virtual terminal path, created by the operating system with the Open Virtual Terminal Path API, QTVOPNVT.		

2	**Read information**	**OUTPUT**	**CHAR(10)**
	Information about the read operation. The characters and their meanings are: **1** The operation code, which gives the server program additional information about OS/400 status and what is expected of the server program. Valid values for this parameter are: **1** Invite **2** Output only **3** Put/get **4** Save display **5** Restore display **6** Read immediate **8** Read display **A** Cancel invite **B** Turn on message light **C** Turn off message light For detailed descriptions of these codes, see "Read Operation Codes" on page 10-6. **2** More data flag. Valid values for this parameter are: **0** There is no more data. **1** More data is available. Issue the read operation again to receive the additional data. This flag is set if the buffer specified on input is not large enough to hold all of the data received from the virtual terminal. **3** Key flag. Valid character values for this parameter are: **0** The Enter key was pressed. **1** The System Request key was pressed. **4-10** Reserved.		
3	**Data buffer**	**OUTPUT**	**CHAR(*)**
	The server program's buffer for receiving data from the virtual terminal. The data is a 5250 data stream. The QTVRDVT API does not lock the data buffer. Thus, other applications should not use the buffer while the API is using it. The data buffer should be large enough to hold the largest display of data expected. If it is not large enough, the more data flag of the read information parameter is set to 1. Additional read requests must be performed, until all the remaining data is received and the more data flag is set back to 0.		
4	**Number of bytes to read**	**INPUT**	**BINARY(4)**
	The number of bytes to read from the data buffer. This number must be smaller than or equal to the size of the data buffer.		
5	**Data received**	**OUTPUT**	**BINARY(4)**
	The amount of data received from the virtual terminal in bytes. If no data is received from the virtual terminal, 0 is returned. Some read operations do not return any data. For graphic work stations, a maximum of 24 576 (24K) bytes of data can be returned.		
6	**Error code**	**I/O**	**CHAR(*)**
	The structure in which to return error information. For the format of the structure, see "Error Code Parameter" on page 10-11.		

Read Operation Codes

The table below describes the operation codes that can be returned on a read request.

Value	Response Name	Description
1	**Invite**	No data is returned from this read request. The operating system and the application are ready to receive data. The server program is expected to follow this with a write operation when data becomes available from the client program.
2	**Output only**	This read request returned some data. The server program should send the data to the client program. However, the operating system is not ready to receive data from the server program. The server program should not request any data from the client program yet. This response usually occurs because an application is performing several put operations to the virtual terminal device. After the last put operation by the application, a put/get operation code is usually returned on the read operation.
3	**Put/get**	Data is returned from this read request and should be sent by the server program to the client program. The operating system is ready to receive data from the server program. The server program should wait for data from the client program.
4	**Save display**	No data is returned from this read request. The operating system expects the server program to obtain the data from the current display and write the data to the virtual terminal. The operating system saves the display for later use, such as returning the display to the server program. The server program must indicate a save-display response on the write operation and send the saved display as data (that is, the saved display must be in the data buffer).
5	**Restore display**	The data returned is a previously saved display. The server program should send the data to the client program.
6	**Read immediate**	No data is returned from this read request. The operating system expects the server program to write data to the virtual terminal. Only data from input fields should be written.
8	**Read display**	No data is returned from this read request. The operating system expects the server program to write data to the virtual terminal. The current display should be written.
A	**Cancel invite**	No data is returned from this read request. The operating system expects the server program to signal the client program to cancel the outstanding invite operation. When it is canceled, the server program must perform a write operation to the virtual terminal and indicate a cancel invite response. Because the response has no data associated with it, the number of bytes to write must be set to 0 for the write operation.

Value	Response Name	Description
B	**Turn on message light**	No data is returned from this read request. A message has been received, and the user should be notified. The server program should signal the client program to turn on a display message indicator light or other indicator.
C	**Turn off message light**	No data is returned from this read request. The display message light or other indicator should be set off.

Write to Virtual Terminal API (QTVWRTVT)

The Write to Virtual Terminal API, QTVWRTVT, writes data from a server program's data buffer to a virtual terminal. You can send one display to the virtual terminal during each write operation. You cannot send partial or multiple displays.

The following error messages are issued by the QTVWRTVT API:

CPF87D4 E	Data sent exceeded the corresponding I/O request
CPF87F2 E	Virtual terminal handle &1 not valid
CPF87F3 E	Data buffer length &1 not valid
CPF87F4 E	Key flag &1 not valid
CPF87F5 E	Operation code response &1 not valid
CPF87F7 E	Parameter value &1 not valid
CPF87F8 E	Unexpected internal system error occurred in program &1

The QTVWRTVT API has these parameters:

QTVWRTVT: Required Parameters			
1	**Virtual terminal handle**	**INPUT**	**CHAR(16)**
	The reference code for the open virtual terminal path, created with the Open Virtual Terminal Path API, QTVOPNVT.		
2	**Write information**	**INPUT**	**CHAR(10)**
	Information about the write operation. The information given in each character is as follows: **1** Key flag. Valid values are: **0** The Enter key was pressed. **1** The System Request key was pressed. The next read operation returns the AS/400 System Request menu. **2** The Attention key was pressed. In this case, the number of bytes to write must be 0. **3** The Test Request key was pressed. **4** The Help-in-Error key was pressed. **2** Operation code. This parameter describes the type of write operation to perform. Valid values and their meanings are: **blank** Put/get **2** Output only **3** Put/get **4** Save display **A** Cancel invite For detailed descriptions of these codes, see "Write Operation Codes" on page 10-9. **3–10** Reserved. These characters must be blank.		
3	**Data buffer**	**INPUT**	**CHAR(*)**
	The server program's buffer containing the data to send to the virtual terminal. The QTVWRTVT API does not lock the data buffer. Thus, other applications should not use the buffer while the API is using it.		

4	**Number of bytes to write**	**INPUT**	**BINARY(4)**
	The number of bytes to write. This number must be smaller than or equal to the size of the data buffer. Valid range of numbers is 0 through 24K. This parameter must be 0 if character 1 of the write information parameter is 2. Some write operations do not write data.		
5	**Error code**	**I/O**	**CHAR(*)**
	The structure in which to return error information. For the format of the structure, see "Error Code Parameter" on page 10-11.		

Write Operation Codes

The table below describes the operation codes that can be used in the write information parameter.

Value	Name	Description
blank	**Put/get**	Data is being sent to the virtual terminal. The virtual terminal server program is ready for input.
2	**Output only**	This write operation is in response to a read request that returned an output-only read operation code.
3	**Put/get**	See the description above.
4	**Save display**	This write operation is in response to a read request that returned a save display read operation code. No data is associated with this write operation; thus, the data buffer length must be set to 0.
A	**Cancel invite**	This write operation is in response to a read request that returned a cancel invite read operation code. No data is associated with this write operation; thus, the data buffer length must be set to 0.

Send Request for OS/400 Function API (QTVSNDRQ)

The Send Request for OS/400 Function API, QTVSNDRQ, sends a request to the operating system to perform a particular function. The requests supported are described in the parameter table below.

The following error messages are issued by the QTVSNDRQ API:

CPF87F2 E	Virtual terminal handle &1 not valid
CPF87F6 E	Request value &1 not valid
CPF87F7 E	Parameter value &1 not valid
CPF87F8 E	Unexpected internal system error occurred in program &1

The QTVSNDRQ API has these parameters:

QTVSNDRQ: Required Parameters			
1	**Virtual terminal handle**	**INPUT**	**CHAR(16)**
	The reference code for the open virtual terminal path, created with the Open Virtual Terminal Path API, QTVOPNVT.		
2	**Request**	**INPUT**	**BINARY(4)**
	The request to be processed by the operating system. Valid binary values are: **1** Cancel Previous Request: Allows the server program to cancel the previous OS/400 request. This is similar to selecting option 2 on the AS/400 System Request menu. **2** Send Break Message: Causes the operating system to issue a break message to the virtual terminal. You can use this to determine whether the virtual terminal is still active.		
3	**Error code**	**I/O**	**CHAR(*)**
	The structure in which to return error information. For the format of the structure, see "Error Code Parameter" on page 10-11.		

Close Virtual Terminal Path API (QTVCLOVT)

The Close Virtual Terminal Path API, QTVCLOVT, closes one or all open virtual terminal paths. To close all open virtual terminal paths, the handle must be set to zero.

The following messages are issued by the QTVCLOVT API:

CPF87F2 E	Virtual terminal handle &1 not valid
CPF87F7 E	Parameter value &1 not valid
CPF87F8 E	Unexpected internal system error occurred in program &1

The QTVCLOVT API has these parameters:

QTVCLOVT: Required Parameters			
1	**Virtual terminal handle**	**INPUT**	**CHAR(16)**
	The reference code for the open virtual terminal path, created with the Open Virtual Terminal Path API, QTVOPNVT. If this parameter is set to zero, all open virtual terminal paths are closed.		
2	**Error code**	**I/O**	**CHAR(*)**
	The structure in which to return error information. For the format of the structure, see "Error Code Parameter."		

VT API Error Reporting

The following sections discuss the standard VT API error code parameters.

Error Code Parameter

Most OS/400 APIs include an error code parameter to return error codes and exception data to the application. The error code parameter is a variable-length structure containing the information associated with an error condition. One field in that structure is an input field; it controls whether an exception is returned to the application or the error code structure is filled in with the exception information. When the input field is nonzero, the rest of the error code structure is filled in with the output exception information associated with the error. When the input field is zero, all other fields are ignored and an exception is returned.

The structure of the error code parameter is as follows:

Offset		Use	Type	Field
Decimal	Hexadecimal			
0	0	INPUT	BINARY(4)	**Bytes provided:** The length of the area provided for the error code, in bytes. The bytes provided must be 0, 8, or more than 8: **0** If an error occurs, an exception is returned to the application to indicate that the requested function failed. **≥8** If an error occurs, the space is filled in with the exception information. No exception is returned.
4	4	OUTPUT	BINARY(4)	**Bytes available:** The length of the exception data available to return, in bytes. If this is 0 (zero), no error was detected.
8	8	OUTPUT	CHAR(7)	**Exception ID:** The identifier for the message for the error condition.
15	F	OUTPUT	CHAR(1)	**Reserved:** A 1-byte reserved field.
16	10	OUTPUT	CHAR(*)	**Exception data:** A variable-length character field containing the insert data associated with the exception ID.

Chapter 11. VT APIs Run-Time Example

To help understand how VT APIs are used, the following example shows how the OS/400, server program, client program, and work station device (display and keyboard) interact when processing a system request.

This example starts with the server program waiting for a response from the client program which is waiting for data from the user (keyboard).

1. System Request processing starts when you press the appropriate System Request key on the work station keyboard.
2. The client program informs the server program that the System Request key has been pressed. The protocol used in this case is unique to the particular implementation of these two programs.
3. The server program performs a call to QTVWRTVT for a write request. The System Request key hit flag must be set for this write request. No data is sent to the virtual terminal at this time.
4. The OS/400 creates a data queue entry for informing the server program that data is available to be read.
5. The server program removes the entry from the data queue by calling QRCVDTAQ and then performs a call to QTVRDVT for a read request. The Cancel Invite operation code is returned. No data is received from the virtual terminal at this time.

   To prevent the client program from sending anymore data (screens), the server program informs the client program that the server program is no longer receiving data from the client program.

   The server program performs a call to QTVWRTVT for a write request. The Operation Code parameter is set to Cancel Invite. No data is sent to the virtual terminal at this time.
6. The OS/400 creates a data queue entry for informing the server program that data is available to be read.
7. The server program removes the entry from the data queue by calling QRCVDTAQ and then performs a call to QTVRDVT for a read request. A Save Screen operation code is returned. No data is received from the virtual terminal at this time.

   The server program gets the current screen. This may require requesting the current screen from the client program.

   The server program performs a call to QTVWRTVT for a write request, sending the current screen to the virtual terminal. The Operation Code parameter must be set to Save Screen.
8. The OS/400 creates a data queue entry for informing the server program that data is available to be read.
9. The server program removes the entry from the data queue by calling QRCVDTAQ and then performs a call to QTVRDVT for a read request. A Put/Get Operation Code is returned. The data read will be the actual System Request menu.

   The server program sends this data to the client program and waits for a response.

10. The client program updates the display with the System Request menu and waits for a response from the user. The resulting response is sent to the server program.

11. The response is received from the client program, and the server program performs a call to QTVWRTTV for a write request, sending the response to the virtual terminal.

    Note: What happens at this point depends on the response to the System Request menu. Additional data may be received from and sent to the virtual terminal. After the response is processed, the following steps occur.

12. The OS/400 creates a data queue entry for informing the server program that data is available to be read.

13. The server program removes the entry from the data queue by calling QRCVDTAQ and then performs a call to QTVRDVT for a read request. A Put Operation Code is returned. The data read is the saved (current) screen that was previously written by the server program to the virtual terminal.

    The server program sends the saved screen to the client program but does not wait for a response.

14. The client program updates the work station display with the saved screen.

15. The OS/400 creates a data queue entry for informing the server program that data is available to be read.

16. The server program removes the entry from the data queue by calling QRCVDTAQ. An Invite Operation Code is returned. Note that no data is received from the virtual terminal at this time.

17. The client program is once again waiting for user data, and the server program is waiting for data from the client program.

Part 3. Appendixes

Appendix A. Additional APIs

Depending on the nature of your application program, it may be necessary or desirable to use one or more of the following APIs:

- user space APIs
- security APIs
- shared folders APIs
- alerts APIs
- data queue APIs

For more information on the user space, security, shared folders, and alerts APIs, see the *System Programmer's Interface Reference*. For more information on the data queue APIs, see the *CL Programmer's Guide*.

Appendix B. 5250 Data Stream and Keystroking Enhancements

This appendix describes enhancements to the AS/400 5250 data stream and keystroke processing support. These enhancements are currently supported by some 5250 work station controllers that attach to the AS/400 system (for example, AS/400 local work station controllers, 5394 remote work station controllers, and in some cases, AS/400 PC Support Work Station Function). The following information applies to the *IBM 5250 Information Display System: Functions Reference Manual* (SA21-9247-6). Page numbers are given below, along with the changes or enhancements that apply to information on those pages.

Changes to the 5250 Functions Reference Manual

Resequencing (Page 2-66)

Note: The controller will check for an endless loop situation and return LUSTAT 00000103 if found.

Orders (Page 2-136)

The Insert Cursor value should be X'13', not X'03'.

Insert Cursor (IC) Order (Page 2-137)

Format: The Insert Cursor value, X'03', should be deleted. The value X'13' is correct.

Restrictions: Add the following restriction: “Row and column values must be less than or equal to the current display screen size (either 24x80 or 27x132).”

Repeat to Address (RA) Order (Page 2-137)

Restrictions: Add the following restriction: “Row and column values must be less than or equal to the current display screen size (either 24x80 or 27x132).”

Set Buffer Address (SBA) Order (Page 2-137 and 2-138)

Restrictions: Add the following restriction: “Row and column values must be less than or equal to the current display screen size (either 24x80 or 27x132).”

Enhancements to 5250 Data Stream Commands and Orders

Input Commands

Read Immediate (Page 2-5): This command has been enhanced to include support for transparent fields. If a transparent Field Control Word (FCW) is found for an input field, the field data is not formatted (for example, nulls are not converted to blanks).

Read Input Fields (Page 2-6): This command has been enhanced to include support for transparent fields. If a transparent FCW is found for an input field, the field data is not formatted (for example, nulls are not converted to blanks).

Read MDT Field (Page 2-7): This command has been enhanced to include support for transparent fields. If a transparent FCW is found for an input field, the field data is not formatted (for example, nulls are not converted to blanks, and trailing nulls are stripped).

Control Characters

Second Byte (Page 2-40): Bit 1 was listed as reserved, but now specifies if the cursor should be moved or not moved at the end of write-to-display processing. If bit 1 is a 0, the cursor continues to be moved to the system IC address on a lock-to-unlock keyboard transition. If bit 1 is a 1, the cursor is not moved.

Field Control Word (FCW)

Transparent Field FCW (Add to Page 2-67): A new FCW has been added to indicate that an input field contains transparent data; that is, the input field contents are sent from the display screen directly to the host at read time with no formatting. Therefore, an entry field can contain any values (X'00' to X'FF'). A transparent field is indicated by a X'84xx' FCW (where xx can be any hex value).

Note: Unpredictable results will occur if a field is defined as both a signed-numeric and transparent field.

Orders

Set Buffer Address (SBA) Order (Page 2-138):

Restrictions: The column size of zero is valid for a row 1/column 1 field.

Start of Field (SF) Order (Page 2-138):

Note: A row 1/column 1 field is defined by an SBA of row 1/column 0, followed by an SF. For a row 1/column 1 input field, the first input-capable position is row 1/column 1. If the SF defines an input field, the screen attribute is not allowed to be nondisplay. Writing of the screen attribute is suppressed for a row 1/column 1 field and the attribute discarded.

New 5250 Data Stream Commands and Orders

Query (Add to Page 2-8)

The Query command is a new input command and is used by the host to obtain information on the capabilities of controller and the 5250 display station. When the controller receives a Query command, the controller sends a Query Reply to the host.

The Query command must follow an Escape (X'04') and Write Structured Field command (X'F3'). The Query command can be sent at any time, as long as the associated device is powered-on and varied-on (LU-LU session is active). The command must be the last command in the SNA chain and CD must be on.

Note: The Work Station Function (WSF) expects this command to be the only command within the data stream. The format of the Query command is as follows:

Byte	Value	Description
0-1	X'0005'	Length of command
2	X'D9'	Command Class
3	X'70'	Command Type - Query
4	X'00'	Flag Byte Bit 0 = B'0', Query Command Bit 1-7 Reserved (set to zero)

When the Query command is received, the controller will send back a Query Reply, defining the controller and the associated device capabilities.

The format of the Query Reply is as follows:

Byte	Value	Description
0-1	X'0000'	Cursor Row and Column (set to zero)
2	X'88'	Inbound Write Structured Field Aid
3-4	X'003A'	Current Length of Query Reply
5	X'D9'	Command Class
6	X'70'	Command Type - Query
7	X'80'	Flag Byte Bit 0 = B'1', Query Reply Bit 1-7 - Reserved (set to zero)
8-9		Controller Hardware Class
	X'0001'	Local Twinax Controller
	X'0061'	Local ASCII Controller
	X'0101'	SDLC/X.21/X.25 Twinax Controller (5394 emulating a 5294)
	X'0103'	SDLC/X.21/X.25 Twinax Controller (5394)
	X'0200'	PC DOS non-DBCS WSF
	X'0300'	OS/2* non-DBCS WSF
	X'0400'	PC DOS DBCS WSF
	X'0500'	OS/2 DBCS WSF
	X'0600'	Other WSF or any other 5250 controller emulator
10-12		Controller Code Level
	X'010300'	For example, Version 1 Rel 3.0
13-28	X'00'	Reserved (set to zero)
29		Device Type
	X'01'	5250 Display or PC emulating a 5250 Display
30-33	C'cccc'	Device Type (e.g. "3180" for "3180" Mod 2)
34-36	C'ccc'	Device Model (e.g. 002 for "3180" Mod 2)
37	X'xx'	Keyboard ID
38	X'xx'	Extended Keyboard ID
39	X'00'	Reserved
40-43	X'xxxxxxxx'	Display Serial Number if From Display
44-45		Maximum number of input fields this display
	X'0100'	Typically = 256 input fields
46-48	X'00'	Reserved (set to zero)

```
Byte       Value              Description

 49     X'xxxxxxxx'     Controller and Display Capability
          Bit 0-1: B'00' - No Row 1/Col 1 support
                   B'01' - Row 1/Col 1 support (nondisplay
                           attribute not allowed at Row 1/Col 0)
                   Other values = reserved
          Bit 2:   B'0'  - No Read MDT Alternate Command Support
                   B'1'  - Read MDT Alternate Command Support
          Bit 3:   B'0'  - No PA1/PA2 support
                   B'1'  - Display and controller have PA1/PA2 support
          Bit 4:   B'0'  - No PA3 support
                   B'1'  - Display and controller have PA3 support
          Bit 5:   B'0'  - No Cursor Select support
                   B'1'  - Display and controller
                   have Cursor Select support
          Bit 6:   B'0'  - No Move Cursor Order support
                   B'1'  - Display and controller have Move Cursor
                           Order support
          Bit 7:   B'0'  - No Read MDT Immediate Alt Command Support
                   B'1'  - Read MDT Immediate Alt Command Support
 50     B'xxxxxxxx'     Controller and Display Capability
          Bit 0-3: B'0001' - 24x80 screen size
                   B'0011' - Capable of 24x80 and 27x132 screen sizes
                   Other values = reserved
          Bit 4:   B'0'     - No Light Pen Support
                   B'1'     - Light Pen Support
          Bit 5:   B'0'     - No Mag Stripe Reader Support
                   B'1'     - Mag Stripe Reader Support
          Bit 6-7: B'00'    - Mono display
                   B'01'    - 5292/3179 style color, including color PCs
                   Other values = reserved
 51     X'00'           Reserved
          Bit 0-4: B'00000' - Reserved
          Bit 5:   B'0'      - No extended primary attribute support
                   B'1'      - Both display and controller have
                               extended primary attribute
                               support (WEA and EA orders)
          Bit 6-7: B'00'   - Reserved
 52     B'xxxxxxxx'     Controller and Display Capability
          Bit 0-2: B'000'   - No DBCS capability
                   B'001'   - Presentation screen DBCS capability only
                   Other Values = Reserved
          Bit 3-7: B'00000' - Reserved
Byte       Value              Description

 53     B'xxxxxxxx'     Controller and Display Capability
          Bit 0-2: B'000'   - No Graphics capability
                   B'001'   - 5292-2 style graphics
                   Other Values = Reserved
          Bit 3-7: B'00000'  - Reserved
54-60   X'00'      Reserved (set to zero)
```

Move Cursor (MC) Order (Add to Page 2-137)

Function: The move cursor (MC) order moves the cursor to the location specified by the two bytes following the order. Byte 1 gives the row address and byte 2 gives the column address. The MC order is useful when the cursor is to be moved without affecting the system IC address. The MC order is unaffected by the WTD control character values including the leave-cursor-flag (CC1 bit 1).

Note: If more than one MC or IC is found in the data stream, the cursor will move to the address specified in the last MC or IC.

Restrictions: A parameter error will be posted when:

- There are fewer than two bytes following the order.
- The row address is 0 or greater than the number of rows on the display screen (24 or 27, depending on screen size).
- The column address is 0 or greater than the number of columns on the display screen (80 or 132, depending on screen size).

Format:

```
Move Cursor Order    Byte 1        Byte 2

      X'14'        Row Address   Column Address
```

Results: The address specified by the MC order is used to move the cursor when the WTD is completed.

Write Extended Attribute (WEA) Order (Add to Page 2-137)

Function: The write extended attribute (WEA) order writes one extended attribute to the current display address of displays that support extended attributes. Extended attributes are written to the display-extended-attribute buffer and do not require a screen position. Two parameter bytes follow the order. Byte 1 defines the attribute type (only extended primary attributes are supported at this time). Byte 2 defines the attribute.

Restrictions: A parameter error will be posted when:

- There are fewer then two bytes following the order.
- The attribute type is not valid
- The attribute type is not supported on this display
- The attribute is not valid

Format:

```
      WEA Order         Byte 1          Byte 2

        X'12'       Attribute Type    Attribute


The Attribute Type for extended primary attributes is X'01'

Valid extended primary attributes are X'00'
and between X'80' and
X'9F' as defined below:

       Bit  0 = Attribute change flag if set
            1 = Reserved
            2 = Reserved
            3 = Column Separator if set
            4 = Blink if set
            5 = Underscore if set
            6 = High Intensity if set
            7 = Reverse Image if set
```

Results: The attribute is written to the display extended attribute buffer and the current display address is not modified. The attributing effect begins at the position of the extended attribute and continues until the next nonzero extended attribute. Extended attributes take precedence over display screen attributes, except when the normal display extended attribute is used (X'80'). The Clear Unit and Clear Unit Alternate commands erase the entire display screen including extended attributes. The erase-to-address order can be used to erase selected extended attributes. The propagate attribute (X'00') can be written with a WEA, which can be used to erase an extended attribute.

Erase-to-Address (EA) Order (Add to Page 2-137)

Function: The erase-to-address order (EA) is similar to the repeat-to-address order. The EA order erases the display screen and extended attributes from the current display address up to the row and column values specified. A minimum of 4 bytes follow the EA order. Byte 1 defines the ending row. Byte 2 defines the ending column. Byte 3 defines the length of remaining data (list of attribute planes to be cleared).

Restrictions: A parameter error will be posted when:

- The row or column values are not valid for the current display screen size (either 24×80 or 27×132)
- The row and column value is less than the current display address
- There are fewer than four bytes following the order
- The attribute type is not valid
- The attribute type is not supported on this display
- The length is not valid

Format:

```
    EA Order  Byte 1   Byte 2   Byte 3       Bytes 4 -

      X'03'     Row     Column   Length   List of Attribute Types
```

The valid Attribute Types are:

```
    X'00'   Display screen (not an extended attribute type, but can
            can be cleared by EA)
    X'01'   Extended Primary Attributes
    X'FF'   Display screen and all supported extended attribute types
```

Results: The display screen and extended attributes are cleared (written to X'00') and the current display address is set to the address specified in the EA plus 1.

Transparent Data (TD) Order (Add to Page 2-137)

Function: The transparent data order (TD) is followed by two length bytes and transparent data. The transparent data is written to the display screen at the current display address; any values (X'00' to X'FF') are allowed in the transparent data. All length values are valid as long as the end of the display screen is not overwritten.

Restrictions: A parameter error will be posted when:

- There are fewer than two bytes following the order
- There are fewer bytes in the data stream then specified in the length field
- Attempting to write beyond the end of the display screen

Format:

```
 Transparent Data Order    Bytes 1 and 2            Bytes 3 and beyond

      X'10'                Length of transparent    Transparent data
                           data (not counting
                           length bytes)
```

Results: The transparent data is written to the display screen and the length of the transparent data is added to the current display address.

New 5250 Keystroke Processing

New Aid Code Generating Keys

PA1 (Add to Pages 2-2 and 2-120): The PA1 aid-generating key has an aid code of X'6C' and does not return input field data.

PA2 (Add to Pages 2-2 and 2-120): The PA2 aid-generating key has an aid code of X'6E' and does not return input field data.

PA3 (Add to Pages 2-2 and 2-120): The PA3 aid-generating key has an aid code of X'6B' and does not return input field data.

New Alphanumeric Key

Field Mark Key (Add to Pages 2-69 and 2-118): The Field Mark key is valid in an input field if the Dup enable/Field Mark enable bit is on in the FFW (byte 1, bit 3). The Field Mark character (X'1E') is then processed like other data characters. If the Field Mark key is pressed in an input field that is not Dup enabled or Field Mark enabled, operator error 0019 is posted.

New Special Keys

Erase EOF (Add to Page 2-125): The Erase End of Field (EOF) key fills all character positions from and including the current cursor position through the end of the field with nulls. The field MDT is also set. The cursor remains at the current position.

Cursor Select (Add to Page 2-126): The Cursor-Select key allows a user to select an input field from the keyboard as if selecting the input field with a Selector Light Pen. The Cursor-Select key is valid only in input fields with a Selector Light Pen FCW (X'8102' or X'8103') otherwise, operator error 0037 is posted. If Cursor Select is keyed in field-exit-required state, operator error 0036 is posted.

Bibliography

If you want more information on a topic while you are using this guide, see the *Publications Guide*, GC41-9678 for related AS/400 publications.

The following publications provide additional information about topics described or referred to in this guide. The manuals are listed with their full title and order number. When these manuals are referred to in text, a shortened version of the title is used.

AS/400 Manuals

- *Communications: Local Area Network Guide*, SC41-0004

  The *Local Area Network Guide* provides information about using an AS/400 system in a token-ring network, Ethernet network, or in a bridged environment.

- *Communications: Management Guide*, SC41-0024

  The *Communications Management Guide* provides information about working with communications status, errors, performance, line speed, and storage requirements.

- *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001

  The *OS/400* Communications Configuration Reference* provides general communications configuration information about lines, controllers, devices, modes, class-of-service, configuration lists, network interfaces, and connection lists.

- *Communications: X.25 Network Guide*, SC41-0005

  The *X.25 Network Guide* provides information about using an AS/400 system in an X.25 packet-switched network.

- *Data Description Specifications Reference*, SC41-9620

  The *DDS Reference* provides detailed descriptions of the entries and keywords needed to describe database files (both logical and physical) and certain device files (for displays, printers, and ICF) external to the user's programs.

- *Database Guide*, SC41-9659

  The *Database Guide* provides information about creating and managing database files.

- *Device Configuration Guide*, SC41-8106

  The *Device Configuration Guide* provides information on how to configure hardware initially and how to change that configuration. Also included is a description of the different keyboard language types. Keyboard language types are specified when using the TELNET function.

- *Guide to Programming Application and Help Displays*, SC41-0011

  The *Guide to Programming Displays* provides information about creating and working with display files. It also provides information about developing online information.

  The method by which AS/400 users can send a 5250 data stream to a display device (user-defined data streams) is described in this manual.

- *IBM 8209 LAN Bridge Customer Information*, SA21-9994

  Describes how to set up and use the 8209 Local Area Network Bridge.

- *Languages: Pascal Reference*, SC09-1210

  The *Pascal Reference* provides information about using the AS/400 Pascal programming language. It includes information you can refer to while using AS/400 Pascal, such as a detailed description of the program structure, declarations, data types, routines, variables, and statements in Pascal.

- *Languages: Pascal User's Guide*, SC09-1209

  The *Pascal User's Guide* provides information about how to use the AS/400 Pascal compiler. The manual explains how to enter, compile, run, and debug AS/400 Pascal programs. It also describes how to use input and output (I/O) function and storage.

- *Languages: Systems Application Architecture* C/400* User's Guide*, SC09-1347

  The *C/400* User's Guide* provides information needed to write application programs or develop programs using the C/400 language.

- *Languages: Systems Application Architecture* AD/Cycle* COBOL/400* User's Guide*, SC09-1383

  The *COBOL/400* User's Guide* provides information needed to design, write, test, and maintain COBOL/400 programs on the OS/400 system.

- *Languages: Systems Application Architecture* AD/Cycle* RPG/400* Reference*, SC09-1349

  The *RPG/400* Reference* provides information needed to write programs for the AS/400 system using the RPG/400 programming language. This manual describes, position by position, the valid entries for all RPG specification forms, and provides a detailed description of all the operation codes. This manual also contains information on the RPG logic cycle, arrays and tables, editing functions, and indicators.

- *Languages: Systems Application Architecture* AD/Cycle* RPG/400* User's Guide*, SC09-1348

The *RPG/400* User's Guide* provides information needed to write, test, and maintain RPG/400 programs on the AS/400 system. The manual provides information on data organizations, data formats, file processing, multiple file processing, automatic report function, RPG command statements, testing and debugging functions, application design techniques, problem analysis, and compiler service information. The differences between the System/38 RPG III, System/38 compatible RPG, and RPG/400 are identified.

- *National Language Support Planning Guide*, GC41-9877

  The *National Language Support Planning Guide* provides information needed to evaluate, plan, and use the AS/400 national language support (NLS) and multilingual capabilities.

- *New User's Guide*, SC41-8211

  The *New User's Guide* provides information about how to sign on and off; send and receive messages, respond to keyboard error messages; use function keys; and control and manage jobs. Also included is a description of keyboard differences.

- *Programming: Control Language Programmer's Guide*, SC41-8077

  The *CL Programmer's Guide* provides a wide-ranging discussion of AS/400 programming topics, including the following:

  - A general discussion of objects and libraries
  - Control language (CL) programming, controlling flow and communicating between programs, working with objects in CL programs, and creating CL programs
  - Predefined and impromptu messages and message handling
  - How to define and create commands and menus
  - Application testing, including debug mode, breakpoints, traces, and display functions

- *Programming: Control Language Reference*, SC41-0030

  The *CL Reference* manual provides a description of the AS/400 control language (CL) and its commands. Each command description includes a syntax diagram, parameters, default values, keywords, and an example.

- *Programming: Reference Summary*, SX41-0028

  The *Programming Reference Summary* provides quick reference information when working with the AS/400 system. This manual contains summaries of information such as system values and OS/400 data description specifications (DDS) keywords.

  If you want to create your own translation tables, this manual discusses creating and editing tables for ASCII line mode.

- *System Concepts*, GC41-9802

  The *System Concepts* provides a general understanding of the concepts related to the overall design and use of the AS/400 system and its operating system. This manual includes general information about AS/400 features such as user interface, object management, work management, system management, data management, database, communications, environments, OfficeVision/400 and PC Support, and system architecture.

- *System Operator's Guide*, SC41-8082

  The *Operator's Guide* provides information on how to respond to error messages and process and manage jobs on the system. Processing jobs includes working with spooled files and finding your printer output.

- *System Programmer's Interface Reference*, SC41-8223

  The *System Programmer's Interface Reference* provides information on how to create, use, and delete objects that help manage system performance, use spooling efficiently, and maintain database files efficiently. This manual also includes information on creating and maintaining the programs for system objects and retrieving OS/400 information by working with objects, database files, jobs, and spooling.

Communications Related Manuals

The following manuals are not specific to the AS/400 system, but do contain helpful communications information.

- *American National Standards Institute/Institute of Electrical and Electronics Engineers 802.2,1985 - Logical Link Control, International Organization for Standardization/Draft International Standard 8802/2.*

- *American National Standards Institute/Institute of Electrical and Electronics Engineers 802.3,1985 - Carrier Sense Multiple Access with Collision Detection, International Organization for Standardization/Draft International Standard 8802/3.*

- *American National Standards Institute/Institute of Electrical and Electronics Engineers 802.3a, b, d, c, 1988 -Supplements to Carrier Sense Multiple Access with Collision Detection American National Standards Institute/Institute of Electrical and Electronics Engineers Standard 802.3*, 1985.

- *American National Standards Institute/Institute of Electrical and Electronics Engineers 802.5,1985 - Token Passing Ring, International Organization for Standardization/Draft International Standard 8802/5.*

- *The International Telegraph and Telephone Consultative Committee, Red Book, Volume VIII -*

Facscicle VIII.3, Data Communications Networks Interfaces, Recommendations X.20 - X.32, VIIIth Plenary Assembly, Malaga-Torremolinos, October 8-19, 1984.

- *Token-Ring Network Problem Determination Guide Kit*, SC30-3374
- *Token-Ring Network Architecture Reference*, SC30-3374
- *3270 Information Display System: 3274 Control Unit Description and Programmer's Guide*, GA23-0061
- *5250 Functions Reference Manual*, SA21-9247

Index

A

API 1-1
APPC
  relationship to user-defined communications 1-4
Application Program Interface 1-1
  QTVOPNVT API parameters 10-1
  QTVRDVT API parameters 10-4
  virtual terminal
    open 10-1
    read 10-4
    run-time example 11-1

C

call accept 3-7
call connected packet 2-49
call request packet 2-12
callable routines 1-2
clear indication packet 4-1
clear request packet 4-1
Client Program 7-1
close connection request 3-5
close virtual terminal 10-11
communications handle 1-3
communications handle, 5-1
communications handle, timer handle
  See handle
connection 1-3, 3-5
connection failure indication 3-8
connection identifier 1-3
connection-oriented 1-3, 3-6
connectionless 1-3, 1-6, 3-32
creating
  virtual controllers 8-3
  virtual devices 8-3

D

data packet 2-24, 3-12
data queues for use with VT APIs 9-1
datagram 1-6
disable 1-3, 2-8
display connection status, inbound routing
  See filter

E

enable 1-3, 2-3
error reporting for VT API 10-11
example
  VT APIs run-time 11-1

F

filter 1-3, 2-10, 3-10
filtering
  LAN based 1-7
  X.25 based 1-7

H

handle 2-1

I

inbound routing 1-7, 2-10, 3-6
incoming call indication 3-25
incoming call packet 2-51, 4-1
input buffer 1-2
interrupt packet 2-24, 3-31, 5-15

L

LAN bridge 3-35
LAN filter format 2-10
limit security officer system value, setting 8-2
link 1-2, 1-3, 2-3, 2-8

O

open connection request 3-5
open virtual terminal 10-1
operation 2-17, 3-3, 3-5
operation codes for read request 10-6
output buffer 1-2

P

PCEP 1-3
PCEP ID 1-3
printer output H-2
printing spooled files H-2
provider connection end point 1-3, 2-17, 3-9
  See also PCEP
  user connection end point 3-9

Q

QAUTOVRT system value for virtual terminal APIs 8-1
QLMTSECOFR system value for virtual terminal APIs 8-2
QTVCLOVT API
  parameters 10-11
QTVOPNVT API
  parameters 10-1
  work station types 10-3

QTVRDVT API
  parameters 10-4
QTVSNDRQ API
  parameters 10-10
QTVWRTVT API
  parameters 10-8

R

read request operation codes 10-6
reset packet 2-29, 2-31, 5-14
RNR packet 3-7
RR packet 3-7

S

send request, virtual terminal 10-10
Server Program 7-1
spooled files H-2
system value
  QAUTOVRT 8-1
  QLMTSECOFR 8-2

T

test frame 1-7
timer handle 2-62

U

UCEP 1-3
UI frame 1-6, 3-32
user connection end point 1-3, 2-17
  See also UCEP
user space 1-2
user-defined communications 1-1
  differences between standard AS/400 communications configuration 1-5

V

virtual terminal
  close parameters 10-11
  description 7-1
  open parameters 10-1
  read parameters 10-4
  send request parameters 10-10
  write parameters 10-8
virtual terminal APIs
  creating programs 8-4
  introduction 7-1
  run-time example 11-1
  security considerations 8-2
  set number of virtual terminals automatically 8-1
  work station types 9-1
VT API error reporting 10-11

W

work station types for virtual terminal APIs 9-1
working with spooled files H-2
write to virtual terminal 10-8

X

XID frame 1-7
X.25 filter format 2-10

Readers' Comments

Application System/400™
System Programmer's Communications
Interface Guide
Version 2

Publication No. SC41-0027-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

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. This form is provided for comments about the information in this manual and the way it is presented.

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.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments
SC41-0027-00

IBM®

Cu
Al

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

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape **Please do not staple** Fold and Tape

SC41-0027-00

Cu
Alc

Readers' Comments

Application System/400™
System Programmer's Communications Interface Guide
Version 2

Publication No. SC41-0027-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

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. This form is provided for comments about the information in this manual and the way it is presented.

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.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments
SC41-0027-00

Cu
Alc

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

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape | **Please do not staple** | Fold and Tape

SC41-0027-00

Cut
Alo

Program Number: 5738-SS1

Printed in Denmark by
Aalborg Stiftsbogtrykkeri A/S

SC41-0027-00