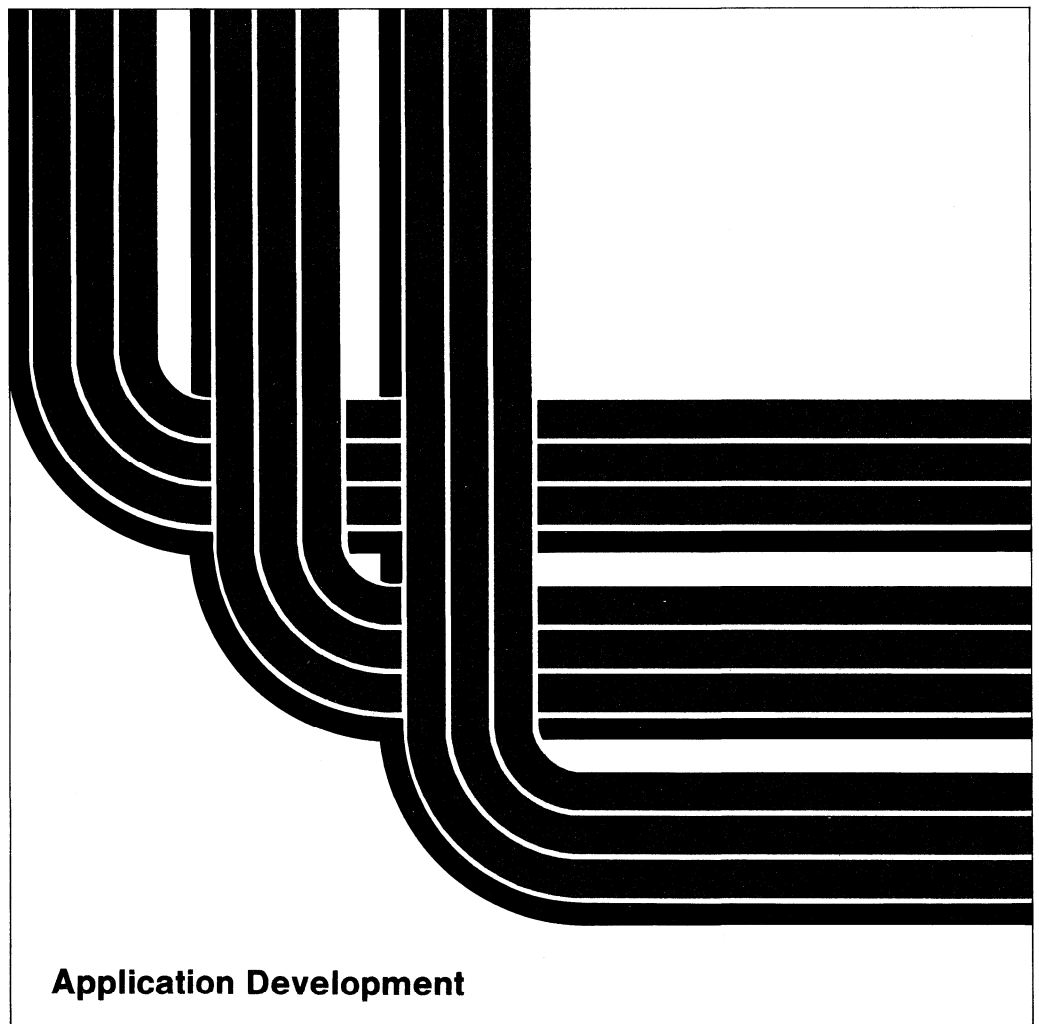


**Communications:
Advanced Program-to-Program
Communications Programmer's Guide**

Version 2





Application System/400

SC41-8189-01

**Communications:
Advanced Program-to-Program
Communications Programmer's Guide**

Version 2

Take Note!

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

Second Edition (September 1992)

This edition applies to the licensed program IBM Operating System/400 (Program 5738-SS1), Version 2 Release 2 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. This major revision makes obsolete SC41-8189-00. 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 Customer Satisfaction Feedback form for readers' comments is provided at the back of this publication. If the form has been removed, you can mail your comments to:

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

or you can fax your comments to:

United States and Canada: 800 + 937 + 3430
Other countries: (+ 1) + 507 + 253 + 5192

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.

© Copyright International Business Machines Corporation 1991, 1992. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii	Special Authority (Security Officer and Service)	3-13
Programming Interface Information	vii	APPC Devices Created by the System	3-13
Trademarks and Service Marks	viii	Application Program Security for ICF	3-13
About This Guide	ix	DDS Security Keyword	3-13
Who Should Use This Guide	ix	When Your Program Initiates the Transaction	3-14
Summary of Changes	xi	System-Supplied Formats Security	3-15
Chapter 1. Introduction to AS/400 APPC		When a Remote Program Initiates the Transaction	3-15
Support	1-1	Application Program Security for CPI Communications	3-15
Support Provided by APPC	1-1	Using CPI Communications with SECURITY(NONE)	3-16
Relationship between APPC and APPN Support	1-1	Using CPI Communications with SECURITY(SAME)	3-16
User-Written Applications	1-1	Chapter 4. Running APPC	4-1
IBM-Supplied Applications	1-2	Vary On and Vary Off Support	4-1
Configuration Requirements	1-3	Vary Configuration On Example	4-2
Communications Lines Supported	1-3	Controlling Modes	4-2
Chapter 2. Configuring APPC	2-1	Start Mode (STRMOD) Command	4-2
Defining the APPC Configuration	2-1	End Mode (ENDMOD) Command	4-3
Configuring an ISDN Network	2-1	Change Session Maximum (CHGSSNMAX) Command	4-4
Line Description	2-1	Displaying the Mode Status	4-6
Controller Description	2-2	Examples of Displaying Mode Status	4-6
Device Description	2-2	Chapter 5. Writing ICF APPC Application Programs	5-1
Mode Description	2-3	Intersystem Communications Function File	5-1
Deleting APPC Configuration Descriptions	2-4	Specifying the Program Device Entry Commands	5-2
Configuration for an APPC Network without APPN	2-4	Communications Operations and Functions	5-3
Chapter 3. APPC Concepts	3-1	Starting a Session Using the Open and Acquire Operations	5-4
APPC Sessions and Conversations	3-1	Starting a Transaction Using the Evoke Function	5-4
Sessions	3-1	Sending Data	5-6
Conversations	3-1	Control-Data Function	5-6
Using the Location Parameters	3-4	Force-Data Function	5-6
Specifying Configurations with APPN(*NO)	3-4	Confirm Function	5-6
Specifying Configurations with APPN(*YES)	3-5	Format-Name Function	5-6
APPC Unit-of-Work Identifier	3-6	Variable-Buffer-Management Function	5-6
APPC Security Considerations	3-6	Receiving Data	5-7
AS/400 Security Levels	3-7	Invite Function	5-7
Physical Security	3-7	Read-from-Invited-Program-Devices Operation	5-7
Session Level Security	3-7	Using the Variable-Buffer-Management Function on Read Operations	5-8
Location Security	3-8		
Validation Tables for Establishing a Session	3-8		
Summary of User IDs Used when the AS/400 System Is the Target System	3-11		
General Security Considerations	3-12		
Invalid Password Attempts via APPC	3-12		

Waiting for a Display File, ICF File, or Data Queue	5-8
Notifying the Remote Program of Problems by Using the Fail Function	5-9
Using Additional Functions and Operations	5-9
Respond-to-Confirm Function	5-9
Request-to-Write Function	5-9
Allow-Write Function	5-9
Timer Function	5-10
Get-Attributes Operation	5-10
Ending a Transaction Using the Detach Function	5-10
Ending a Session	5-10
Release Operation	5-10
End-of-Session Function	5-10
Close Operation	5-11
Using Response Indicators	5-11
Receive-Confirm	5-11
Receive-Fail	5-11
Receive-Turnaround	5-12
Receive-Detach	5-12
Receive-Control-Data	5-12
Using Input/Output Feedback Areas	5-12
Using Return Codes	5-13
Mapping between ICF Operations and Functions and LU Type 6.2 Verbs	5-13
Mapping of LU Type 6.2 Verbs to ICF Operations and Functions	5-13
Example ICF Operations and Functions Mapped to LU Type 6.2 Verbs	5-15
Flow Diagrams	5-18
A Note about the Flow Diagrams	5-18
Flow Diagram for Inquiry Applications Using ICF Operations and Functions	5-19
Flow Diagram for Inquiry Applications Using LU Type 6.2 Verbs	5-20
Chapter 6. Writing APPC Application Programs Using CPI Communications	6-1
Description of Communications Side Information	6-1
Managing the Communications Side Information	6-2
Using Program Calls	6-4
Program Calls Supported by the AS/400 System	6-5
Using Pseudonyms When Writing Applications	6-6
Mapping of CPI Communications Calls to ICF Operations and Functions	6-7
Example ICF Operations and Functions Mapped to CPI Communications Calls	6-8
Flow Diagram for Inquiry Applications Using CPI Communications Calls	6-11
Return Codes for CPI Communications	6-14

Chapter 7. Application Considerations for ICF	7-1
General Considerations	7-1
Open and Acquire Operation Considerations	7-1
WAITFILE Considerations	7-2
Output Considerations	7-2
Input Considerations	7-2
Confirm Considerations	7-3
End-of-Session, Release, and Close Considerations	7-4
Prestart Jobs Considerations	7-4
Trace ICF Communications Considerations	7-4

Chapter 8. Application Considerations for CPI Communications	8-1
General Considerations	8-1
Performance Considerations	8-1
Prestart Job Considerations	8-2
Trace CPI Communications Considerations	8-2

Appendix A. ICF Language Operations, Data Description Specifications Keywords, and System-Supplied Formats	A-1
Language Operations	A-1
Language Operations Supported	A-1
Data Description Specifications Keywords	A-2
System-Supplied Formats	A-3

Appendix B. Sense Data and Return Codes	B-1
Program Start Request Errors	B-1
SNA Sense Data	B-4
Return Codes	B-4
Major Code 00	B-5
Major Code 02	B-9
Major Code 03	B-14
Major Code 04	B-18
Major Codes 08 and 11	B-19
Major Code 34	B-20
Major Code 80	B-23
Major Code 81	B-27
Major Code 82	B-30
Major Code 83	B-37
CPI Communications Return Codes	B-45

Appendix C. Implementation of the LU Type 6.2 Architecture	C-1
AS/400 System Implementation of Control Operator Verbs	C-1
Change-Number-of-Sessions Verbs	C-1
Session-Control Verbs	C-2
LU Definition Verbs	C-2
ICF Implementation of the LU Type 6.2 Architecture	C-3
Specifying the Resource Parameter	C-3

Mapped Conversation Verbs	C-3	RPG/400 Local Program for Inquiry	
Basic Conversation Verbs	C-12	Applications (Example 3)	E-36
Miscellaneous Verbs	C-19	RPG/400 Remote Program for Inquiry	
Mapping of LU 6.2 Return Codes to ICF		Application (Example 3)	E-44
Return Codes	C-20		
LU 6.2 Conversation Verb Option Sets		Appendix F. Example Programs for	
Used by the AS/400 System	C-23	Applications Using the CPI Comm	
LU 6.2 Control-Operator Verb Option Sets		Application Program Interface	F-1
Used by the AS/400 System	C-26	Objects Used by Program Examples	F-1
		Communications Side Information Object	
Appendix D. APPC Configuration Examples	D-1	(T8189CSI)	F-1
Configuration Example for a Two-System		Display File Object (T8189DSP)	F-2
APPC Network (without APPN, Switched)	D-1	Database File Object (T8189DB)	F-2
Configuring System B (Los Angeles)	D-3	C/400 Local Program for Inquiry Applications	
Configuration Example for a Two-System		(Example 1)	F-2
APPC Network (without APPN,		C/400 Remote Program for Inquiry	
Nonswitched)	D-3	Applications (Example 1)	F-11
Configuration Example for X.21 Short-Hold		COBOL/400 Local Program for Inquiry	
Mode	D-6	Applications (Example 2)	F-18
Configuring the New York System	D-6	COBOL/400 Remote Program for Inquiry	
Configuring the Los Angeles System	D-8	Application (Example 2)	F-27
Configuration Example for Programs		RPG/400 Local Program for Inquiry	
Communicating on the Same System	D-9	Applications (Example 3)	F-35
		RPG/400 Remote Program for Inquiry	
Appendix E. Example Programs for		Application (Example 3)	F-47
Applications Using the ICF Application			
Program Interface	E-1	Bibliography	G-1
Objects Used by Program Examples	E-1	Planning and Installation Manuals	G-1
ICF File Object (T8189ICF)	E-1	Customer and System Operation	
Display File Object (T8189DSP)	E-2	Manuals	G-1
Database File Object (T8189DB)	E-3	AS/400 Communications Manuals	G-1
C/400 Local Program for Inquiry		AS/400 Programming Manuals	G-1
Applications (Example 1)	E-3	PC Support/400 Manuals	G-1
C/400 Remote Program for Inquiry		Communications Architectures	G-1
Applications (Example 1)	E-12	SAA CPI Communications	G-2
COBOL/400 Local Program for Inquiry		Index	X-1
Applications (Example 2)	E-21		
COBOL/400 Remote Program for Inquiry			
Application (Example 2)	E-29		

Notices

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

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

This publication could contain technical inaccuracies or typographical errors.

This publication may refer to products that are announced but not currently available in your country. This publication may also refer to products that have not been announced in your country. IBM makes no commitment to make available any unannounced products referred to herein. The final decision to announce any product is based on IBM's business and technical judgment.

Changes or additions to the text are indicated by a vertical line (|) to the left of the change or addition.

Refer to the "Summary of Changes" on page xi for a summary of changes made to the OS/400 Licensed Program and how they are described in this publication.

This publication contains small programs that are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

Programming Interface Information

Common Programming Interface (CPI) Communications provides an application programming interface for applications that use program-to-program communications. The interface uses the SNA LU 6.2 architecture to perform the following services:

- Establish a conversation.
- Send and receive data.
- Exchange control information.
- End a conversation.
- Notify a partner program of errors in the communication.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*), used in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

ACF/VTAM	Application System/400
AS/400	C/400
COBOL/400	FORTRAN/400
IBM	NetView
Operating System/400	OS/2
OS/400	Personal Computer/AT
Personal Computer XT	Personal System/2
RISC System/6000	RPG/400
SAA	Series/1
SQL/400	System/370
System/390	Systems Application Architecture
VTAM	XT
400	

About This Guide

This guide describes the advanced program-to-program communications (APPC) support provided by the AS/400 system. It is intended for the application programmer responsible for developing application programs that use the APPC support.

Included in this guide are application program considerations, configuration requirements and commands, problem management for APPC, and general networking considerations.

Although this guide does contain some information about systems other than an AS/400 system, it does *not* contain all the information that the other systems may need to communicate with an AS/400 system using APPC. For complete information for a particular remote system type, refer to that system's documentation.

You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide* provides information on all the manuals in the AS/400 library. For a list of related publications, see the "Bibliography" on page G-1.

Note: Throughout this guide, when references are made to other publications, only the short-

ened version of the title is used. The "Bibliography" contains a list of related manuals with full titles and order numbers.

Who Should Use This Guide

This guide is intended for the application programmer responsible for developing application programs that use the APPC support.

You should have knowledge of general communications concepts. AS/400 communications concepts are covered in the *System Concepts* 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/Education course in the communications module. The Discover/Education course is separately orderable.

It is also assumed that you have read the following or have the equivalent knowledge:

- *Operator's Guide*
- *ICF Programmer's Guide*
- *SAA* CPI Communications Reference*

Summary of Changes

This guide has been updated to include the following:

- Support providing the ability to process user control data.

- Support providing the ability to defer an EVOKE function.

- Support providing the ability to process overflow data (the excess data that occurs when a program is sent a data record longer than its capacity to receive data).

Chapter 1. Introduction to AS/400 APPC Support

The AS/400* **advanced program-to-program communications (APPC) support** is the AS/400 system implementation of the Systems Network Architecture (SNA) logical unit (LU) type 6.2 and node type 2.1 architectures. Throughout this guide, APPC is used generically to refer to the application program interface and general support supplied by the AS/400 system to the LU 6.2 and node type 2.1 architectures.

Support Provided by APPC

The APPC support handles all of the SNA protocol requirements when your system is communicating with a remote system using the LU type 6.2 and node type 2.1 architectures. You can connect your system to any other system that supports the APPC program interface.

Relationship between APPC and APPN Support

The AS/400 APPC support handles the protocol needed for communicating between an application program running on your AS/400 system and an application running on a remote system. The protocol, which is defined by the SNA LU 6.2 architecture, consists of a set of verbs that are common to the local and remote systems in a network. However, the way in which each system provides a program interface to the verbs may differ.

The AS/400 system provides these three program interfaces:

- The intersystem communications function (ICF) file interface
- The Common Programming Interface (CPI) Communications call interface
- The Customer Information Control System (CICS) file interface

In ICF, the LU 6.2 verbs are implemented using data description specifications (DDS) keywords and system-supplied formats. In CPI Communications, the LU 6.2 verbs are implemented using CPI Communications calls. In CICS/400 support, the LU 6.2 verbs are implemented using EXEC CICS commands. Refer to "Mapping between ICF Operations and Functions and LU Type 6.2 Verbs" on page 5-13 for a list of LU 6.2

verbs and their corresponding DDS keywords or system-supplied formats. Refer to "Mapping of CPI Communications Calls to ICF Operations and Functions" on page 6-7 for a mapping of CPI Communications calls and ICF operations and functions. The *CPI Communications Reference* manual provides a mapping between LU 6.2 verbs and CPI Communications calls. For a list of the EXEC CICS commands used to control an APPC conversation, see the *CICS/400 Administration and Operations Guide*.

The APPC support also handles networking functions, and allows peer systems in a network to start and end sessions without a controlling host system.

The AS/400 **advanced peer-to-peer networking (APPN) support** is an enhancement to the node type 2.1 architecture, and provides additional networking functions such as searching distributed directories, dynamically selecting routes, routing of intermediate sessions, creating and starting remote locations, and routing data using transmission priorities. The *APPN Guide* provides details about the APPN support.

User-Written Applications

APPC provides both an interactive and batch communications interface between user-written application programs on local and remote systems. An AS/400 system can start programs on a remote system, or a remote system can start programs on an AS/400 system.

An AS/400 application using the APPC support can:

- Write to or read from an ICF file
- Access the CPI Communications call interface, which is part of the Systems Application Architecture* (SAA*) framework
- Use the EXEC CICS commands

If you use ICF, you can write application programs using C/400*, COBOL/400*, SAA FORTRAN/400* (FORTRAN/400), or RPG/400*. Chapter 5 of this guide and the *ICF Programmer's Guide* contain information about writing applications using ICF files.

If you use CPI Communications, you can write application programs in SAA C/400, COBOL/400, FORTRAN/400, Procedures Language 400/REXX (an SAA Procedures Language), RPG/400, and Cross System Product (CSP) (an SAA application generator). Chapter 6 of this guide and the *CPI Communications Reference* manual contain information about writing applications using the CPI Communications call interface.

If you use CICS, you can write application programs using COBOL/400. For more information about writing CICS application programs, see the *CICS/400 Application Programming Guide*.

Note: An APPC conversation cannot be used by both the System/38 environment and the AS/400 system operating environment. An escape message is sent to an application that is attempting to open an ICF file or to allocate a CPI Communications conversation in the AS/400 operating environment and to open a communications file or a mixed device file in the System/38 environment for the same APPC conversation.

An application can use a combination of ICF, CPI Communications, and CICS/400 interfaces. However, when communicating with another program, an APPC application cannot use combinations of ICF operations or functions and CPI Communications calls or EXEC CICS commands on the same conversation.

IBM-Supplied Applications

APPC also provides the communications capabilities for the following:

- **Distributed data management (DDM)**, a function of the operating system that allows an application program or user on one system to access database files stored on remote systems. Refer to the *DDM Guide* for additional information.
- **SNA distribution services (SNADS)**, an asynchronous distribution service that defines a set of rules to receive, route, and send electronic mail in a network of systems. Refer to the *Distribution Services Network Guide* for additional information.
- **SNA pass-through**, a function that allows SNA data to be passed from source secondary applications or devices to target primary applications. SNA pass-through supports LU session types 0 through 3.

Refer to the *Remote Work Station Guide* for additional information.

- **Display station pass-through**, a function that allows a user to sign on to one system (an AS/400 system, System/38, or System/36) from another system (an AS/400 system, System/38, or System/36) and use that system's programs and data. Refer to the *Remote Work Station Guide* for additional information.
- **File transfer support**, a function of the operating system that moves file members from one system to another by using asynchronous, APPC, or BSCEL communications support. Refer to the *ICF Programmer's Guide* for additional information.
- **PC Support/400**, which provides system functions to an attached personal computer. Refer to the *PC Support/400 DOS Installation and Administration Guide* or the *PC Support/400 OS/2 Installation and Administration Guide* for additional information.
- **Alert support**, which provides the support for handling alerts on the AS/400 system. Alerts, which the system detects, are used to report a problem or an impending problem to the network operator. Refer to the *Alerts and DSNX Guide* for additional information.
- **Electronic customer support**, a function of the operating system that allows a customer to access the question-and-answer (Q & A) function; problem analysis, reporting, and management; IBM product information; and technical information exchange. Refer to the *Device Configuration Guide* for additional information.
- **Customer Information Control System (CICS/400) support**, which allows user-written application programs to process transactions entered at remote work stations. CICS/400 provides the CICS environment with CICS-specific functions, and the AS/400 environment with functions that are not CICS-specific. Refer to the *CICS/400 Administration and Operations Guide* or the *CICS/400 Application Programming Guide* for additional information.

For each of these IBM-supplied applications you must configure the APPC support on each of the systems in the network. Chapter 2 in this manual and the *OS/400* Communications Config-*

uration Reference contain additional information concerning configurations.

Configuration Requirements

You configure your system to use the APPC support with the configuration menus or the control language commands supplied with the AS/400 system. These configuration requirements are discussed in Chapter 2 and in the *OS/400* Communications Configuration Reference*, which also has a description of the communications configuration process.

Information about the communications equipment you may need, and the IBM-supplied APPC functions that are supported can be found in the *Network Planning Guide*.

Communications Lines Supported

The number of configurations that can be varied on at the same time depends on the size of your system and the type of communications adapters attached. Each line used by APPC can be one of the following (all the lines on your system and throughout the network do not have to be the same type):

- **IDLC (ISDN data link control)**¹ switched or non-switched
- Synchronous data link control (SDLC) point-to-point switched (manual answer, automatic answer, manual call, or automatic call)
- SDLC point-to-point nonswitched
- SDLC multipoint nonswitched
- X.25 packet switched data network supporting both permanent virtual circuit and switched virtual circuit connections
- Token-ring network
- Ethernet network

In addition to supporting the line types mentioned, APPC allows the sharing of these lines as follows:

- If an APPC communications type is configured to use *primary* SDLC support, it can also share a multipoint line with finance support, retail support, and with remote work station support. They can all be active on the multipoint line at the same time.
- If an APPC communications type is configured to use *secondary* SDLC support, it can share a line with the SNA version of **remote job entry (RJE)**², **3270 device emulation for SNA**³, **SNA upline facility (SNUF)**⁴, **distributed systems node executive (DSNX)**⁵ support, and **distributed host command facility (DHCF)**⁶ support. They can all be active on the line at the same time.
- If an APPC communications type is configured to use X.25 support, then multiple remote systems can be active at the same time. The APPC support can share the X.25 line with finance support, retail support, remote work station support, the SNA version of RJE, 3270 device emulation for SNA, SNUF, DSNX support, DHCF support, and **Transmission Control and Protocol/Internet Protocol (TCP/IP)**⁷.

¹ ISDN is the acronym for integrated services digital network, a CCITT recommendation that defines an interface to a network that can carry voice, data, and image over the same communications line. IDLC is an asynchronous, balanced data link protocol used between two systems to exchange information over an ISDN network.

² RJE allows a user to submit a job from a display station on the AS/400 system to a System/370-type host system.

³ 3270 device emulation allows an AS/400 system to appear as a 3274 Control Unit in an SNA network.

⁴ SNUF allows an AS/400 system to communicate with CICS/VS and IMS/VS application programs on a host system.

⁵ DSNX is a function of the operating system that receives and analyzes requests from the NetView* Distribution Manager licensed program on a host system. If the request is directed to the system that receives it, the request is processed on that system or on a personal computer directly attached to that system. If the request is intended for a different system, it is routed toward its destination.

⁶ DHCF is a function of the operating system that supports the data link between a System/370 terminal using an AS/400 application in an HCF (Host Command Facility) environment.

⁷ TCP/IP is a set of vendor-independent communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

- If an APPC communications type is configured to use a local area network, then multiple remote systems can be active at the same time. The APPC support can share the local area network with finance support, remote work station support, the SNA version of RJE, 3270 device emulation for SNA, SNUF, DSNX support, DHCF support, and TCP/IP.
- If an APPC communications type is configured to use IDLC, the APPC support can share the IDLC line with remote work station support, the SNA version of RJE, 3270 device emulation for SNA, SNUF, DSNX support, and DHCF support.

Chapter 2. Configuring APPC

This chapter describes the commands used for configuring APPC.

When using APPC configuration commands, you can enter the commands in one of two ways:

- Using the command prompt. Enter the command and press F4 (Prompt). A prompt menu is shown for the command.
- Using direct entry. Enter the command and its parameters following the syntax described in the *CL Reference*.

The following is a brief introduction to the communications descriptions you need to configure before you can use the APPC support. A complete description of these and related commands are contained in the *APPN Guide*, the *OS/400* Communications Configuration Reference*, and the online help.

Defining the APPC Configuration

The AS/400 system with APPC support allows you to create and store many APPC configuration descriptions on the system. Each configuration description name must be unique.

An APPC configuration consists of a line, an APPC or host controller, an APPC device, and a mode description. If your program is communicating with a partner program on the same system, you do not need to configure a line description.

If you are using an integrated services digital network (ISDN), you must also create a connection list and network interface description.

Configuring an ISDN Network

Using an ISDN network requires that you configure a connection list and network interface description before the line, controller, device, and mode descriptions. The *ISDN Guide* contains more information about configuring an ISDN network.

1. Connection Lists

A **connection list** contains configuration information that is required by the system to

manage how calls are sent or received across an ISDN network. Information about connection lists is referred from the line description or from one of the related controller descriptions. The following commands are used when working with connection lists and with the entries in the connection lists:

- Create Connection List (CRTCNLL)
- Change Connection List (CHGCNLL)
- Display Connection List (DSPCNLL)
- Delete Connection List (DLTCNLL)
- Work with Connection Lists (WRKCNNL)
- Add Connection List Entry (ADDCNNLE)
- Change Connection List Entry (CHGCNNLE)
- Rename Connection List Entry (RMCNNLE)
- Remove Connection List Entry (RMCNNLE)
- Work with Connection List Entry (WRKCNNLE)

2. Network Interface Description

The **network interface description** contains information required by the system to communicate with, and to describe the system interface to, an ISDN network. The following commands are used when working with the network interface descriptions:

- Create Network Interface Description (ISDN) (CRTNWIISDN)
- Change Network Interface Description (ISDN) (CHGNWIISDN)
- Delete Network Interface Description (DLTNWID)
- Display Network Interface Description (DSPNWID)
- Work with Network Interface Description (WRKNWID)

Line Description

The line description describes the physical line connection and the data link protocol to be used between the AS/400 system and the network. The following commands are used to create or change line descriptions:

- Create Line Description (IDLC) (CRTLINIDLC)

- Change Line Description (IDLC) (CHGLINIDLC)
- Create Line Description (SDLC) (CRTLINS DLC)
- Change Line Description (SDLC) (CHGLINS DLC)
- Create Line Description (X.25) (CRTLINX25)
- Change Line Description (X.25) (CHGLINX25)
- Create Line Description (Token-Ring Network) (CRTLINTRN)
- Change Line Description (Token-Ring Network) (CHGLINTRN)
- Create Line Description (Ethernet) (CRTLINETH)
- Change Line Description (Ethernet) (CHGLINETH)
- Work with Line Descriptions (WRKLIND)

Controller Description

The controller description describes adjacent systems in the network. Certain parameters on the Create Controller Description (APPC) (CRTCTLAPPC) or Create Controller Description (HOST) (CRTCTLHOST) commands dictate how the local system treats the adjacent system. The following commands are used to create or change controller descriptions:

- Create Controller Description (APPC) (CRTCTLAPPC)
- Change Controller Description (APPC) (CHGCTLAPPC)
- Create Controller Description (HOST) (CRTCTLHOST)
- Change Controller Description (HOST) (CHGCTLHOST)
- Work with Line Descriptions (WRKLIND)

Defining Controller Descriptions for ISDN Networks: If you are configuring an ISDN network, use the value *IDLC or *X25 for the link type (LINKTYPE) parameter for these commands.

Defining Controller Descriptions for Programs Communicating on the Same System: If you are configuring an APPC controller for use by source and target programs that are communicating with each other on the same system, use the value *LOCAL for the link type (LINKTYPE) parameter for the CRTCTLAPPC command.

Note that source and target programs communicating with each other on the same system do not require a line description. All APPC

program functions are supported, and the fact that data is not actually leaving the system is transparent to the APPC application programs.

The source and target programs must use different APPC device descriptions, but share the same controller description.

The two device descriptions must be created on the same controller with partner local and remote location names. For an example, see “Configuration Example for Programs Communicating on the Same System” on page D-9.

Defining APPC Controller Descriptions for Use with SNA Pass-Through Support: SNA pass-through support allows a user signed on to an AS/400 system attached to an SNA backbone network to pass through the AS/400 system and access primary applications on a System/370 or System/390 host computer.

For each logical SNA pass-through session, you must configure two lines, an APPC controller, a host controller, and a minimum of two devices. One of these devices is attached to the communications line flowing toward the source device (downstream) and one is attached to the communications line flowing toward the target application (upstream). You can group upstream devices logically by configuring them in a configuration list. You can then request to communicate with the first available device on the list rather than requesting a single device which may be unavailable.

Activate the SNA pass-through session by configuring and varying on SNA pass-through (*SNPT) devices attached to the AS/400 system through an APPC controller.

For more information about SNA pass-through configuration, refer to the *Remote Work Station Guide*.

Device Description

The device description describes the characteristics of the logical connection between two locations in the network (the local location and the remote location), and contains information about the device, or logical unit, that is attached to the system. The device description allows you to specify the RMTLOCNAME (the name of the remote location associated with the device

description) and the LCLLOCNAME (the name assigned to the local location). The remote location name, when combined with the remote network ID, identifies a remote location to your local AS/400 system. This name must match one of the local location names specified in the configuration definition on the remote system. The local location name, when combined with the local network ID, identifies your local system to a remote system. This name must match one of the remote location names specified in the configuration definition on the remote system.

Note: No two devices on the same controller can have the same combination of names specified for the following: remote location name, local location name, and remote network identifier.

The device descriptions exist only at the end points (the local location and the remote location) of a session. There are no device descriptions for intermediate sessions at a network node. When using the APPC support, device descriptions are not created automatically, as they are when you are using the APPN support (that is, APPN(*YES) is specified in the controller description). Therefore, you must create a device description for each end point in your network. The following commands are used to create or change device descriptions:

- Create Device Description (APPC) (CRTDEVAPPC)
- Change Device Description (APPC) (CHGDEVAPPC)
- Work with Device Descriptions (WRKDEVD)

To display the status of the network interface, line, controller, and device descriptions for APPC, use the Work Configuration Status (WRKCFGSTS) command. This command allows you to determine if the line is available for use. The Retrieve Configuration Status (RTVCFGSTS) command can also be used from a CL program to determine device status. Refer to the *Communications Management Guide* for more information about these commands.

Defining Device Descriptions for Programs Communicating on the Same System: If you are configuring an APPC device for use by source and target programs that are communicating with each other on the same system, you must create two device descriptions on the same controller, and create partner local and remote location

names. For an example, see “Configuration Example for Programs Communicating on the Same System” on page D-9.

Mode Description

The mode description describes the session characteristics and number of sessions that are used to negotiate the allowed values between the local and remote location. A mode with the same name must exist at both end points (the local location and the remote location) of a session. The mode need not exist for an intermediate session except at the network node server for a low-entry networking node.

Important System Name Information

Use caution when you choose names that use the special characters #, \$, and @. These special characters might not be on the remote system’s keyboard. The names that may be exchanged with remote systems include the following:

- Network IDs
- Location names
- Mode names
- Class-of-service names
- Control point names

Five predefined modes are shipped with the AS/400 system:

- BLANK, the default mode name specified in the network attributes when the system is shipped. Using this mode results in a mode name of 8 blanks (hex 40). This mode is equivalent to the *BLANK session group name on a System/36 and to the *BLANK mode name parameter of the Add Device Mode Entry (ADDDEVMODE) command on a System/38.
- #BATCH, a mode tailored for batch communications.
- #BATCHSC, which is the same as #BATCH except that the associated class-of-service description requires a data link security level of at least *PKTSWTNET (packet switched network).
- #INTER, a mode tailored for interactive communications.
- #INTERSC, which is the same as #INTER except that the associated class-of-service description requires a data link security level

of at least *PKTSWTNET, (packet switched network).

Every local location on your local system will use the values specified in the mode to negotiate session limits with every remote location. After negotiating the session limits, the limits are kept between each local location and remote location.

Note: For a single session device, the values for the mode parameters MAXSSN, MAXCNV, LCLCTLSSN, and PREESTSSN are not used. The values for MAXSSN, MAXCNV, LCLCTLSSN, and PREESTSSN come from the device description for configurations not using APPN and from the APPN remote location list for configurations using APPN. Refer to the *OS/400* Communications Configuration Reference* for the values defined in the IBM-supplied mode descriptions and for descriptions of the parameters and values associated with mode descriptions. The following commands are used to create or change mode descriptions:

- Create Mode Description (CRTMODD)
- Change Mode Description (CHGMODD)
- Work with Mode Descriptions (WRKMODD)

Certain parameters for the configuration commands can only be changed when the configuration description is varied off. Refer to the *OS/400* Communications Configuration Reference* to determine if you must vary off the configuration description to make changes. If the configuration description is varied off to make changes, you must vary on the configuration description after the changes are made. This permits you to use the new attributes for your session.

Deleting APPC Configuration Descriptions

To delete descriptions associated with an APPC configuration, use the following commands:

- Delete Network Interface Description (DLTNWID)
- Delete Line Description (DLTLIND)
- Delete Controller Description (DLTCTLD)
- Delete Device Description (DLTDEVVD)
- Delete Mode Description (DLTMODD)
- Delete Connection List (DLTCNNL)

Configuration for an APPC Network without APPN

An APPC network without APPN is one in which the systems in the network are **low-entry networking nodes**. A low-entry networking node is a node that implements the node type 2.1 architecture without the APPN extensions. If you are configuring an AS/400 system, for example, and you specify APPN(*NO) on the controller description, then the AS/400 system is considered to be a low-entry networking node. In this case, you do not need to configure transmission group and node characteristics or class-of-service descriptions. This information is only needed when using the APPN support. For information on configuring APPN networks, see the *APPN Guide*.

An example of an APPC network without APPN is shown in Figure 2-1.

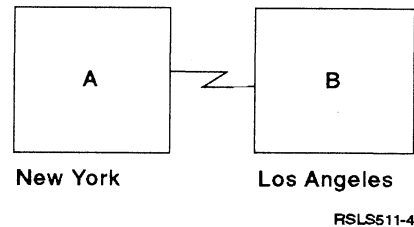


Figure 2-1. Two-System APPC Network (without APPN)

In this figure, System A and System B are considered to be the low-entry networking nodes.

In this network you must configure an APPC device description for all of the locations that will be communicating between the two systems.

Note: A low-entry networking node may participate in an APPN network by using the services of an attached network node server. However, the user at a low-entry networking node must configure all the remote locations with which communications will occur as if those locations existed at the network node server. The ability for a low-entry networking node to use the APPN network varies depending on how the system is configured.

Chapter 3. APPC Concepts

The following topics in this chapter for APPC apply to both the ICF file interface and the CPI Communications call interface.

APPC Sessions and Conversations

A session gives an application program on one system a logical connection to a remote system. Starting a session is analogous to establishing a telephone connection between two offices. Once the application program is connected to a session, it needs to establish a conversation, which enables communications to occur with the target program. A conversation is analogous to the dialogue that two persons exchange over a telephone connection. This section describes how the concept of sessions and conversations are implemented on the AS/400 system.

Sessions

In an APPC communications environment, no single system is responsible for the control of all sessions. Instead, control of the sessions is distributed among the systems in the network. For example, when two AS/400 systems are connected using APPC, each system may initiate (bind) and end (unbind) sessions. Those sessions for which your system is responsible are called locally controlled sessions. Sessions for which the remote system is responsible are called remotely-controlled sessions.

The application program does not directly connect to a session. It may not specify a particular session over which it wishes to communicate. Instead, when an application program asks to establish a session to a remote location, the APPC support selects an available session from the mode and builds a connection between the application program and the session. A locally controlled session is selected if one is available. If no locally controlled session is available, then an available remotely-controlled session is used. When CPI Communications is used to request the session and *return_control* is CM_IMMEDIATE, only available, active, locally controlled sessions are selected.

The MAXSSN parameter of the Create Mode Description (CRTMODD) command is used to specify a limit on the total number of sessions.

Conversations

Conversations may be **synchronous** or **asynchronous**. In a synchronous conversation, the two programs directly communicate with one another. A conversation is asynchronous when the sending program completes its transmission of data and ends its conversation before the receiving program receives all the data. This can be due to the delay inherent in the communications line or the AS/400 data buffers, or the specific protocols used to send the data.

When a conversation becomes asynchronous, its temporary connection with the session is broken. An application program with an asynchronous conversation can continue to receive data from the local AS/400 system data buffers until all the data is returned to it, but it can no longer transmit because it has been disconnected from the session. In the meantime, the session may be used by another application program. There can be more application programs running, or more conversations, than there are sessions.

The MAXCNV parameter of the Create Mode Description (CRTMODD) command is used to specify a limit on the total number of conversations that can be run at the same time for a particular mode to avoid overuse of AS/400 system resources. By specifying a value for the number of conversations for the SNGSSN parameter on the Create Device Description (APPC) (CRTDEVAPPC) command, you can control the number of conversations for single-session APPC devices.

Conversations between type 6.2 logical units always use **half-duplex flip-flop protocols**, in which they alternate sending requests to one another. Parameters in the bind identify one of the logical units as **first speaker**, or the speaker that begins in the send state. The other logical unit begins in the receive state. The first speaker allows the other logical unit to become the sender by setting the change direction indicator (located in the request header of the last

request unit sent) from sender to receiver status. The two logical units can continue to switch between send and receive state until they deactivate the session.

Once an APPC conversation has started, the link between the two application programs is also called a **transaction** for ICF.

The AS/400 APPC support allows programs to interact with the LU type 6.2 support using mapped or basic conversations. The following sections describe the difference between the two conversation types.

Mapped Conversations: When using mapped conversations, the application program is responsible for, and only sends or receives, the user data portion of the **communications data stream**¹. The system performs all the processing of the **architected**² length (LL) and **general data stream identifier (GDS ID)** that precede all user data sent to, or received from, the remote system. The *Architecture Logic for LU Type 6.2* contains more information about the length (LL) and the general data stream (GDS).

Mapped Conversations for ICF: The CNVTYPE parameter on an ICF device entry command specifies the conversation type. To use a mapped conversation, both the source and target programs must agree. The source program must be specified as *SYS for the CNVTYPE parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. The target program, however, may specify either *SRCPGM or *SYS.

Note: If you use mapped conversations, the Variable Buffer Management (VARBUFMGMT) keyword (see page 5-6) is the only ICF keyword that is not supported.

Mapped Conversations for CPI Communications:

CM_MAPPED_CONVERSATION is the default for the *conversation_type* characteristic set by the Initialize_Conversation (CMINIT) call. A source program can also set the *conversation_type* to

CM_MAPPED_CONVERSATION by issuing a Set_Conversation_Type (CMSCT) call before allocating a conversation with the Allocate (CMALLC) call. A target program cannot set the conversation type, but can determine the *conversation_type* by issuing the Extract_Conversation_Type (CMECT) call.

Basic Conversations: When using basic conversation support, the user program is responsible for the integrity of certain parts of the communications data stream that the program never sees when using mapped conversations.

When using basic conversation support, the architected length (LL) and general data stream identifier (GDS ID) bytes that precede all user data in the communications data stream are received from, and presented to, the user program along with the user data. The system assumes that the first 2 bytes in the output buffer contain the LL when the program issues a write operation. The system also assumes that the first 2 bytes of data received from the remote system are the LL and passes them, as received, to the user program when it issues a read operation.

The system makes no assumptions and has no requirements for the presence of the GDS ID bytes. Your application program is responsible for the content of the GDS ID bytes to ensure the user program functions properly with the remote APPC system or program.

When using basic conversation support, you should have an in-depth knowledge of the other system or program capabilities and requirements, and you should also understand the LU type 6.2 general data stream (GDS) structure. A GDS variable (a record) is made up of one or more of the architected structures (LL segment) shown in Figure 3-1. More than one LL segment is used when the GDS variable cannot be contained in a single LL segment because of system or program limitations and is referred to as a concatenated GDS variable.

¹ In SNA, the communications data stream is made up of a structured field, called the general data stream, followed by the user data. The general data stream consists of a length (LL), which is defined as the first two bytes of the structured field, and a general data stream identifier (GDS ID), which is defined as the next two bytes following the length field that identifies the GDS-defined format of the data.

² Architected refers to the specification set forth for the LU type 6.2 protocol.

Figure 3-1. General Data Stream Structure

Byte	Name	Description
0,1	LL	<p>Architected length. Bit 0 is the concatenation bit, which is ignored by the system. It can be used by the program or the remote system or program. According to the architecture, if the bit is off, this segment is the last, or only, part of the GDS variable. If the bit is on, the GDS variable does not complete with this segment and one or more LL segments follow.</p> <p>Bits 1 through 15 contain the binary value of the total length of the current GDS variable or, if concatenation is being used, the length of the current portion of the GDS variable. The value must include the 2 bytes of the LL itself.</p>
2,3	GDS ID	<p>Identifier. The system treats these bytes as user data when using basic conversation support. However, the other APPC system or program may depend on these bytes containing meaningful information.</p> <p>According to the architecture, if this is the first or only LL segment in a GDS variable, then the identifier value is contained in bytes 2 and 3. If this is an LL segment that follows an LL segment that had the concatenation bit set on, then these 2 bytes contain the first 2 characters of user data that is to be concatenated to the data previously received. The identifier is provided only in the first LL segment of concatenated GDS variables.</p>
4 - n	User data	Unarchitected user data.

Basic Conversations for ICF: The CNVTYPE parameter on an ICF device entry command specifies the conversation type. To use a basic conversation, both the source and target programs must agree. The source program must specify *USER for the CNVTYPE parameter on the ADDICFDEVE, CHGICFDEVE or OVRICFDEVE command. The target program, however, may specify either *SRCPGM or *USER.

Note: Only the FMTNAME DDS keyword is not supported when using basic conversations.

This section discusses basic conversations in which the VARBUFMGMT function is *not* used. For information about basic conversations in which the VARBUFMGMT function *is* used, refer to “Variable-Buffer-Management Function” on page 5-6.

The AS/400 system treats the LL as user data. If VARBUFMGMT is not used, it also uses LL as a length. That is, these bytes are passed, as is, to or from the user buffer along with the other data on input and output operations. When you use keywords that cause the system to examine specific locations in the data stream (RECID for example), you must note that the system considers the first 4 bytes (LL and GDS ID) to be a part of the record. Therefore, if you expect the system to examine the first byte of unarchitected user data, you should specify the fifth position.

In addition, the system considers each LL segment to be a separate record, regardless of whether the concatenation bit is set.

When issuing input or output operations on a basic conversation, the system ignores all data in the input or output buffer beyond the length specified by the LL. Your programs do not need to blank fill the output buffer beyond that length. On read operations, the system does not pad the buffer with blanks when the LL is smaller than the record format length. Any bytes in the buffer beyond the length specified by the LL are considered to be data that cannot be predicted (for example, data left from a previous operation). If a RECID DDS keyword is specified with a position beyond that defined by the LL, it matches only if the compare value for the RECID DDS keyword is a blank (hex 40).

Basic Conversations for CPI Communications:

To use the basic conversation support, your source program should set the *conversation_type* to CM_BASIC_CONVERSATION by issuing a call to Set_Conversation_Type (CMSCT). This must be done before the conversation is allocated with an Allocate (CMALLC) call. Target programs cannot set the conversation type, but can use Extract_Conversation_Type (CMECT) to determine the *conversation_type*.

Using the Location Parameters

For a user or IBM-supplied application using APPC, the remote location name, device, local location name, remote network ID, and mode parameters that can be specified on various system commands are used to determine the link to the remote system. These parameters can be specified, for example, on an ADDICFDEVE or OVRICFDEVE command, on a Create Communications Side Information (CRTCSI) or Change Communications Side Information (CHGCSI) command, or in a DDM file. Refer to the documentation for the product you are using for information on specifying these parameters. If any of the parameters are not specified (either because you did not specify them or the command does not allow the parameter to be specified) the following defaults are used:

- Remote location name: No default; parameter is required.
- Local location name: *LOC
- Device: *LOC
- Remote network ID: *LOC
- Mode: *NETATR

Remote location names are used to make application programs independent of communications devices by allowing a remote location name to be used to access a remote communications resource. A remote communications resource is represented on the AS/400 system as one or more device descriptions; therefore, a remote location name is a logical name that is used to select which particular device description or descriptions should be used.

Because the remote location name is the basis for determining which device description should be used, it is necessary to specify the remote location name when creating any device description that can be used by APPC. These device descriptions must be manually created in all cases except when using advanced peer-to-peer networking (APPN) support. In general, APPN does not require that remote location names be configured because the system searches the network and finds the location at the remote system where the location name was defined as a local location name. For the special cases when remote locations need to be configured for APPN, it is done through a config-

uration list. In either case, the system creates device descriptions containing the remote location name when APPN is being used.

Once the AS/400 system has determined that the application requires APPC, then the processing depends on whether APPN support is being used or not.

Specifying Configurations with APPN(*NO)

More than one device description can contain the same remote location name and other matching parameter values. This results in a match for the local location name and remote network ID. Special values for resolving these parameters are as follows:

- If a device is specified as *LOC, any device name will match.
- If a local location name is specified as *LOC, any local location name will match.
- If a remote network ID is specified as *LOC, any remote network ID name will match.
- If a local location name is specified as *NETATR, the value used is retrieved from the network attributes.
- If a remote network ID is specified as *NETATR, the value used is retrieved from the network attributes.

You have the choice of letting the system select a device description based on the three values (remote location name, local location name, remote network ID), or requesting a specific device description. If you let the system select the device description, the APPC device descriptions are searched alphabetically for all descriptions that match the remote location name, local location name, and remote network ID parameters. Of the devices that match these parameters, the system chooses the first device that it finds with the best status.

The best status is determined in the following order:

- A device with a status of active is chosen (if available)
- A device with a status of varied on or a switched device with a status of vary on pending is chosen (if available)

- A device with a status of recovery pending or a nonswitched device with a status of vary on pending is chosen (if available)

If no device is available that satisfies any of the conditions listed above, a device is still chosen but an error message and return code will also be given to the application.

Note: Single session devices that are in use are not chosen if devices with unused sessions are available.

This allows devices that are varied off or in recovery to be passed over in favor of devices that are likely to have a session (for example, a device that is already actively communicating with the remote system). An error occurs if the system does not find a device that has a status of active, varied on, varied on pending, or recovery pending.

If you request a specific device description, then it is selected if all of the following conditions are true:

- The device description contains the specified remote location name
- The local location name and remote network ID result in a match
- The device is not in a varied-off status.

Requesting a specific device description is recommended in the following cases:

- When device descriptions containing the same remote location name, local location name, and remote network ID are attached to more than one controller description. Selecting the device description allows you to control which line and controller are used.
- When a specific logical unit on the host system is required by the AS/400 application program.

If a device description is not found, the request to establish a session fails. If the device description is selected, the mode parameter is then processed. The selected device description must have the mode configured as a valid mode. The mode must also be started and, once started, have available sessions. If none of these conditions are met, the request to establish a session fails. Special values for the mode are resolved as follows:

- *NETATR: Value to use for the mode is retrieved from the system network attributes.

- BLANK: Mode is represented in the network as 8 blank (hex 40) characters.

Specifying Configurations with APPN(*YES)

The device, local location name, and remote network ID parameters are first processed as follows:

- The value specified for the device description (DEV parameter) is ignored for APPN processing. If APPN must automatically create (configure) a device description, APPN will assign a name to the description.
- If the local location name (LCLLOCNAME parameter) is *NETATR or *LOC, then the local location name is retrieved from the network attributes.
- If the remote network ID (RMTNETID parameter) is *NETATR, *LOC, or *NONE, then the remote network ID is retrieved from the network attributes (the LCLNETID parameter).

The system then attempts to find the remote location within the APPN network (specified by the remote network ID parameter) and determines a route to the remote location. The device description selected is one that matches the remote location name, local location name, and remote network ID and that is attached to the controller description representing the first connection of the calculated route. If a device description is selected, then it will be activated (varied on) if necessary. If a device description does not exist, then one is automatically created (configured) and activated.

If an appropriate device description cannot be selected, the request to establish a session fails. If a device description is selected, the MODE parameter is then processed. The specified MODE must be configured on the system. If the mode is configured and it is not already attached to the selected device description, it is attached and started automatically by the system. If the mode is already attached, it is then started if it has not already started.

If the mode is not configured, cannot be attached to the device, cannot be started, or has no sessions available, the request to establish a

session fails. Special values are processed as follows:

- ***NETATR:** The value to use for the mode is retrieved from the system network attributes.
- **BLANK:** The mode is represented in the network as 8 blank (hex 40) characters.

APPC Unit-of-Work Identifier

The **unit-of-work identifier**³ is an option of the LU type 6.2 architecture that is supported by IBM products that use APPC. The AS/400 system supports both sending and receiving of the unit-of-work identifier. This identifier is built by the system whenever it sends a program start request so that the system can track a distributed transaction involving resources on various systems. The AS/400 system always sends the unit-of-work identifier when an application using ICF issues an evoke function, or when an application using CPI Communications issues the allocate call. If a program start request is received by the AS/400 system from another system and the program start request does not carry the unit-of-work identifier, the AS/400 system builds a unit-of-work identifier and associates it with the job being started on the AS/400 system.

The unit-of-work identifier can be found in three places within the system. When the APPC job is running, the unit-of-work identifier can be displayed by entering the Display Job (DSPJOB) command for that job. When an APPC job ends, the unit-of-work identifier is printed on the AS/400 job log. To use user-defined auditing, a message (CPI9803) is sent to the history file. This message contains the job name and the unit-of-work identifier. Even after the jobs have ended, the unit-of-work identifier and job name of all APPC jobs can be determined by examining the history file using the Display Log (DSPLOG) command.

The unit-of-work identifier can be used when you need to know what jobs on two or more systems are involved in a transaction. The AS/400 system displays the unit-of-work identifier when option 1 is selected from the Display Job display.

For distributed transactions that involve an AS/400 system to another AS/400 system, the unit-of-work identifier can be used to determine the job names on each AS/400 system that are involved in the transaction. For example, if two AS/400 systems, one in Chicago and one in Denver, are running an APPC transaction, the following commands can be used to determine the unit-of-work identifier so that a problem can be resolved:

1. The AS/400 operators at both locations enter the Work with Configuration Status (WRKCFGSTS) command to display all active APPC jobs within each system.
2. Assuming that both operators know the name of the device description that represents the other AS/400 system, option 5 (Work with job) can be selected to display each APPC job. Select option 1 from the Display Job display to display the unit-of-work identifier. The unit-of-work identifier shown at both systems will be the same.

If the operator does not know the device description that represents the other AS/400 system, then all APPC jobs running under all APPC device descriptions must be displayed to find the unit-of-work identifier that matches the job on the remote system. You can also use the Display Log (DSPLOG) command with MSGID(CPI9803) specified.

APPC Security Considerations

There are four aspects of security for an AS/400 system, a System/38, and a System/36 communicating with each other using APPC and APPN:

- **Physical security** surrounding the systems, modems, communication lines and display stations that can be configured in the line description and used in the route selection process
- **Location security** that verifies the identity of other systems in the network
- **User ID security** that verifies the identity and rights of users to issue commands on their local system and remote systems

³ A unit of work is the amount of processing that is started directly or indirectly by a program on the source system. The unit-of-work identifier is a unique label assigned to a unit of work.

- **Resource security** that controls user access to particular resources, such as confidential databases

Only security for communications or multiple systems management is discussed in this section. Security needs to be consistent across all the systems in a network if intersystem access is to be controlled and yet not unnecessarily restricted.

Location, user ID, and resource security are only possible if the system security level is set at an appropriate level. The AS/400 security levels are explained in more detail in “AS/400 Security Levels.”

IBM-supplied application programs and user-written application programs have different security implementations that must be understood. An important issue is what user ID is used when communicating with the remote system. Default user IDs can be provided for some applications but not for others, and customers must decide the requirements that determine the use of default user IDs.

AS/400 Security Levels

An understanding of the base security functions of the AS/400 system is necessary when discussing network security.

When the system is using level 10 security, AS/400 APPC connects to the network as a non-secure system. The AS/400 system does not validate the identity of a remote system when a session is being established, and does not require transaction security on incoming program start requests.

Note: For level 10, security information configured for the APPC remote location (location password and secure location in the device description or remote location list) is ignored and is not used when a session or conversation is being established.

Security is dependent on both systems. If the AS/400 system is the remote system and is using level 20 or above, AS/400 APPC connects to the network as a secure system. The sending system can then provide both session and application level security functions.

The security level is a system value (QSECURITY) that can be set with the Change System Values (CHGSYSVAL) command or from the configuration menu as part of an initial program load (IPL) of the system. The change takes place after the next IPL of the system.

For more information on security levels, see the *Security Reference* manual.

Physical Security

You are responsible for the physical security of your system when you specify *NONE for the location password (LOCPWD) parameter during APPC configuration. In this case, the AS/400 system does not validate the identity of a remote system when a session is being established. However, you can still use application level security if the remote system supports it, for example, if the remote system is an AS/400 system with security level 20 or above. Refer to “Application Program Security for ICF” on page 3-13 and “Application Program Security for CPI Communications” on page 3-15 for additional information.

Session Level Security

Session level security or security on the bind is achieved by specifying a password on the LOCPWD parameter during configuration. The AS/400 system uses the password to validate the identity of the remote system when a session is being established. The password must match the password specified on the remote system or the connection is not allowed. Not all systems support session level security. For example, Series/1 RPS version 7.1 and CICS/VS release 1.6 do not. If the remote system does not support session level security, you must specify LOCPWD(*NONE) to establish the connection, and provide the necessary physical security.

If the location passwords specified for two systems are the same (or null), then the sessions are considered secure and they allow location security.

See Figure 3-2 for a list of the null password implementations for the AS/400 system, System/38, and System/36.

Figure 3-2. Null Password Implementations

System	Parameter	Null Value
AS/400 System	Remote Config. List Loc. Password or LOCPWD parameter	*NONE
System/38	CRTDEVD SYSVLPWD	*NONE
System/36	SECEDIT COMM Location Password	*NULL

Both the System/38 and the AS/400 system (with level 20 or above) have a location security manager with a null password but the System/36 does not. This means that a location must be explicitly defined with the System/36 SECEDIT COMM procedure if session-level security is to exist between a System/36 and another location (System/36, System/38 or AS/400 system). This is the same as if a location is not defined with SECEDIT COMM or if password security is not active so that no locations can be defined with SECEDIT COMM.

A non-null-location password enhances security because there is greater certainty about the identity of the remote system. This is especially important in X.25 networks and switched communications.

However, in a large network an equally large number of passwords may be required if any-to-any connectivity is required. To simplify the password management function, the pass-

Figure 3-3. Secure Location Parameter

System	Secure Location Parameter	Secure Value	Nonsecure Value
AS/400 System	SECURELOC on device description or secure location in Remote Location Configuration List	*YES	*NO
System/38	SECURELU on CRTDEVD	*YES	*NO
System/36	Require User Password on SECEDIT COMM	N	Y

Validation Tables for Establishing a Session

Figure 3-4 and Figure 3-5 show the allowed security levels for session establishment (or BIND) that result from all the combinations of location passwords in a peer network. When a session fails to be established (the BIND fails),

word can default to a value of null without reducing application program security.

Location Security

If the location passwords on a local and remote system match, a location can specify the amount of security information required from the other location for each remotely initiated APPC conversation. Then applications do not need to send passwords with user IDs. Network administration is simplified because user passwords need not be the same on all systems or need not be required in each APPC application.

A **secure location** means that a target system has acknowledged that security on the source system is acceptable. Because the target system accepts the security arrangements of the source system, the source program can then send an indicator called an already verified indicator (AVI) with the user ID.

A target system is a **nonsecure location** in one of two ways. If the target system is level 10, it cannot accept security information. If the target system is level 20 or higher, it can accept either a user ID and a password or no security information at all, in which case the target system uses a default user ID, if defined.

See Figure 3-3 for how a secure location is specified on the target AS/400 system, System/38 and System/36.

no APPC communications is possible between these two systems. When the remote system allows incoming conversation-level security parameters, your system may use the signed-on user ID; a user ID and password; or *NONE. When the remote system does not allow incoming conversation-level security parameters, your system may not send the signed-on user ID.

Figure 3-4. Establishing a Session between an AS/400 System and a System/38, System/36, or another AS/400 system.

Remote System/Security	Local AS/400 Security Level ¹		
	10	> 10 (*NONE Specified)	> 10 (PASSWORD Specified)
AS/400 System			
Security = 10	Security not accepted	Security accepted ³	Fails
Security > 10 (*NONE ²)	Security not accepted	Security accepted	Fails
Security > 10 (PASSWORD ²)	Fails	Fails	Security accepted if passwords match
System/38			
*NONE	Security not accepted	Security accepted	Fails
PASSWORD	Fails	Fails	Security accepted if passwords match
System/36			
No password	Security not accepted	Security accepted ⁴	Fails
*NULL	Security not accepted	Security accepted	Fails
PASSWORD	Fails	Fails	Security accepted if passwords match
Notes:			
<ol style="list-style-type: none"> 1 The information under each security level indicates the effect of each security level on session establishment (BIND). 2 These are values for the LOCPWD parameter on the CRTDEVAPPC command. 3 Security accepted indicates that the incoming conversation-level security parameters (user ID and password or user ID and already verified indicator (AVI) without a password) are allowed. 4 Security is accepted but the AVI is never sent. 			

Figure 3-5. Establishing a Session between a System/36 or System/38 and Another System/36 or System/38

Remote System/ Password	System/38 ¹ Location Password		System/36 ¹ Location Password		
	*NONE	PASSWORD	No Password	*NULL	PASSWORD
System/38					
*NONE	Security accepted ²	Fails	Security not accepted	Security accepted	Fails
PASSWORD	Fails	Security accepted if passwords match	Fails	Fails	Security accepted if passwords match
System/36					
No password	Security not accepted	Fails	Security not accepted	Security not accepted	Fails
*NULL	Security accepted	Fails	Security not accepted	Security accepted	Fails
PASSWORD	Fails	Security accepted if passwords match	Fails	Fails	Security accepted if passwords match
<p>¹ The information under each password location indicates the effect of each password location on session establishment (BIND).</p> <p>² Security accepted indicates that the incoming conversation-level security parameters (user ID and password or user ID and AVI without a password) are allowed.</p>					

Summary of User IDs Used when the AS/400 System Is the Target System

Figure 3-6 shows where user IDs are extracted when the AS/400 system is the target system and the session establishment is one of the following:

- Security not accepted
- Security accepted and secure location (AVI) is specified
- Security accepted and secure location (AVI) is not specified

Figure 3-6. Summary of User IDs Used when the AS/400 System is the Target System

Session Establishment	Applications using Sub-system Default User ID ¹	Applications using Source Sign-on User ID ²	Applications using User ID and Password Sent in Program Start Request ³
Security not accepted ⁴	DDM ⁵ SNADS ⁶ DSPT ⁷ FTS ⁸ User-written program	Not applicable	Not applicable
Security accepted ⁴ and SECURELOC(*YES)	AS/400 DSPT S/38 DSPT SNADS User-written program (ICF only)	AS/400 DSPT S/36 DSPT DDM FTS User-written program	AS/400 DSPT User-written program (ICF only)
Security accepted ⁴ and SECURELOC(*NO)	DDM SNADS DSPT User-written program FTS	Not applicable	AS/400 DSPT User-written program (ICF only)

Notes:

- 1 The subsystem default user ID is used when no security information is received on the program start request. This is denoted as SECURITY(NONE).
- 2 The currently signed on user ID is sent as the verified user ID on the program start request. This is denoted as SECURITY(SAME).
- 3 The user ID sent in the program start request is denoted as SECURITY(PGM).
- 4 A security level (QSECURITY) system value of 10 is not secure. A value of 20 or greater is secure.
- 5 Distributed data management
- 6 SNA distribution services
- 7 Display station pass-through
- 8 File transfer support

User-written applications using ICF can choose to send any security information, depending on the coding. They can do the following:

- Send the calling user's ID and AVI if the remote system accepts security and it has specified SECURELOC(*YES).
- Send application coded user ID and password if they are secure or not.
- Send no security information.

It is not possible to access a user's password from the security file. It must be either coded in an application or prompted for in an application.

Security information sent by programs using CPI Communications is dependent on the secure location parameter of the target system. If a location is specified as a secure location, then the calling user's ID and AVI is automatically sent by CPI Communications. Otherwise, no security information is sent. AS/400 CPI Communications does not support SECURITY(PGM).

General Security Considerations

When application program security is used, you should avoid sending and receiving passwords unnecessarily on start requests. The system initiating the transaction should validate the user ID. This removes the possibility of having a password intercepted during transmission.

In any network with communications between secure systems, it is very important that the person responsible for network security ensure that each user has a unique user ID throughout the network. When local verification of the user ID is performed, only the user ID is passed between secure systems on the network. Any user on a remote system having a user ID the same as a local user ID has access to the local system with all of the authority of the local user. For example, a user on a system in New York should not have the same user ID as a user on a system in Los Angeles if the New York user is starting jobs on both systems.

Invalid Password Attempts via APPC

It is possible for a password that is not valid to be entered by a remote user; this can mean that an error occurred or an attempt was made to break security. You should consider preventing any further use of the device until the situation is understood.

A password that is not valid can be sent by a remote user when the remote system sends a program start request. When a program start request is rejected, the system sends message CPF1269 to the QSYSOPR message queue. The system identifies the error as a password that is not valid. The message will be directed to the QSYSMSG message queue if it exists.

A system administrator may set a limit on the number of consecutive password attempts that are not valid for a given display device. When this limit is reached, the device is then varied off. The limit is set with the system value QMAXSIGN on both the AS/400 system and System/38 and with the SECDEF command on the System/36.

On the System/36, this limit also applies to display station pass-through users from remote locations. If a display station pass-through user fails to sign on within the limit set on a

System/36, the System/36 logically disables that user's location. This means that no user from that location is allowed to start a program on the System/36. Message SYS-8437 is placed in the history file when each attempted program start request is rejected. Users currently active are not affected. This situation is cleared by disabling the subsystem (not just the location) and re-enabling it on the target System/36. This could affect many users on many systems, some of which are not using the System/36 (except as an intermediate APPN node).

Sign-ons exceeding the limit on the System/38 and the AS/400 system are not handled the same way on the System/36. Both vary off the virtual display station pass-through device. Other APPC users are not affected and the situation is cleared by varying on the virtual display station pass-through device. If there are other virtual display station pass-through devices defined, then subsequent users can select them until they are all varied off or in use. Other APPC applications such as file transfer support (FTS) or distributed data management (DDM) are not affected by the QMAXSIGN value.

When an AS/400 device is varied off, message CPF1397 is issued. The message might also be used as an alert because it may indicate that someone has attempted to break security, or that someone has forgotten a password and needs assistance to set it again.

Some customers may wish to view the QHST log for security violation messages. This can be done by using the Display Log (DSPLOG) command and specifying message IDs that relate to security. These include CPF1107, CPF1120, CPF2234, CPF1269, and CPF1397. The output can be sent from an output queue to the central location for analysis using object distribution.

Users can sign on to one AS/400 system, System/38 or System/36 more than once with the same profile. If this does not suit a customer's environment, you can limit the user profile to one sign-on by use of the QLMTDEVSSN system value (*SYSVAL) for the LMTDEVSSN parameter on the CRTUSRPRF or CHGUSRPRF command.

When the employees at distributed sites do not receive education about the security aspects of their system, then central security control is

needed. Distributed site employees need to keep the security officer informed of transfers or leaves so user profiles can be removed. Central control also provides consistent implementations across the network so that one site does not ignore security.

Special Authority (Security Officer and Service)

Although user profiles that grant security officer and service authorities give those users special authority, one of the following conditions must be true in order for received program start requests to be accepted from a user with security officer or service authority.

- The user must be explicitly authorized to the AS/400 APPC device description through which the program start request was received.
- The user must have created the APPC device description.
- The QLMTSECOFR system value is set to zero, which indicates security officer and service authorities are not to be limited.

For example, if an AS/400 system receives a program start request with a user ID and password that selects a user profile that has as its special authority, security administrator or service authority, and this user ID has not been explicitly granted authority to the APPC device description, then the program start request is rejected.

This aspect of system security is consistent with the way work station security is done for the security officer and service user profiles. Anyone who has object management authority for an APPC device description can grant authority for the APPC device description to the security officer or service user IDs.

If the security officer or service user creates an APPC device description, the security officer or service user (like anyone else who creates device descriptions) is explicitly authorized to the device description.

When a device is authorized to all users (*ALL), the user IDs that have security administration or service authority are **not** included. This allows the security officer or service user to specify the

APPC device description from which security officer or service functions can be performed.

APPC Devices Created by the System

Devices created automatically by the system are created with an object authority of *CHANGE. All users, except those with security officer or service authorization, have object management authority. You can change the object authority of these devices by using the Change Device APPC (CHGDEVAPPC) command and the appropriate security commands.

Application Program Security for ICF

This section discusses for application program security considerations for ICF.

DDS Security Keyword

The DDS SECURITY keyword, *n reserved-word* | *literal* | *field-name-1* | *&field-name-1* [.3.], allows you to include security information along with the evoke function. Any record format for which the SECURITY DDS keyword is specified (or implied by being specified at the file level) must also have the EVOKE DDS keyword specified on the format or at the file level. If the SECURITY keyword is specified without the EVOKE keyword, the file being defined is not created.

The SECURITY keyword can be specified at the file and record level. Option indicators are allowed, and are required if the keyword is specified more than once for a format. All security specifications must satisfy the requirements of the target system.

The *n* parameter identifies the security subfield being described and can be specified as

- 1 for a profile ID
- 2 for a password
- 3 for a user ID

The values that can be specified for the security fields are

- *reserved-word*:
 - *USER specifies that the user's profile name on the local AS/400 system is to be used as the security field.

- *NONE specifies that the security field is not to be supplied. *NONE must be specified if the requested program is to run using the default user. When this option is selected, only the length (2) and the type value are sent to the remote system. If you want to send blanks as the security field, you must specify it as a literal or use a field-name.

For example, if SECURITY(3 *NONE) is specified, the security sent to the remote system would be a value put into code for a user ID, but no user ID would be included. If the remote system allows a default user ID, it would accept this type of security request.

- *'literal'*: a literal value (up to 10 characters) that contains the needed security information.
- *field-name (or &field-name)*: (this value is not allowed if the SECURITY DDS keyword is defined at the file level) the name of a field within the record format that contains the needed security information. The field defined cannot be longer than 10 characters.

The SNA LU 6.2 architecture security values are specified using the DDS security keyword as follows:

- SECURITY(NONE) is done by not specifying the SECURITY DDS keyword or by specifying SECURITY(3 *NONE).
- SECURITY(SAME) is done by specifying SECURITY(3 *USER). If a profile ID was received to start this job, it is passed on whenever an EVOKE is done by specifying SECURITY(3 *USER).
- SECURITY(PGM) is done in the following manner:
 - DDS SECURITY keyword (user ID, password, and profile ID are all allowed) specified with the EVOKE keyword
 - For system-supplied formats, the following can be specified in the source program data buffer:
 - Positions 9 through 16, password
 - Positions 17 through 24, user ID

When Your Program Initiates the Transaction

When you specify security information with the evoke function, you establish **application program security**. This provides security parameters (profile ID, password, or user ID) that the remote system may require. You can use application program security if the remote system supports it, even if security validation is not done when the session is being established (physical security is being used). For example, if the remote system is an AS/400 system with security level 20 or above active and *NONE is configured for LOCPWD.

The SNA LU 6.2 architecture supports sending three levels of security when allocating a conversation:

SECURITY(NONE)

No security information is sent.

SECURITY(SAME)

The ID of the signed on user is sent. If a profile ID was received to start this job, the profile ID will also be sent.

SECURITY(PGM)

The security information is supplied by the application program.

For evoke functions issued on the AS/400 system, the security information you specify is handled as follows:

- If the remote system does not have a security manager (for example, AS/400 level 10 security), no security information is sent as a result of the evoke function; therefore, the security level is SECURITY(NONE).
- If the remote system does have a security manager (for example, AS/400 security levels 20, 30, and 40), then one of the following occurs:
 - No security information is sent; therefore, the security level is SECURITY(NONE).
 - Security information supplied by the application program on the evoke function is sent; therefore, the security level is SECURITY(PGM).
 - Security information that the application program requests APPC to supply (by specifying SECURITY(3 *USER) on the

DDS EVOKE keyword or specifying a user ID of all blanks for a system-supplied format evoke function) is sent as follows:

- If the remote system allows local verification of the user (for example, the remote system is an AS/400 system with SECURELOC(*YES) configured), APPC sends the sign-on user ID of the user and the already verified indicator to indicate that the local system has already verified the user ID; therefore, the security level is SECURITY(SAME). The user ID must exist on the remote system.
- If the remote system does not allow local verification of the user (SECURELOC(*NO)), APPC does not send any security information; therefore, the security level is SECURITY(NONE).

System-Supplied Formats Security

For system-supplied formats, security is specified in the data buffer of the application program, as follows:

- Positions 9 through 16 is the password
- Positions 17 through 24 is the user ID

Note: Profile ID cannot be specified when using system-supplied formats.

The values that can be specified for the security fields are:

- *'literal'*: a literal value (up to 8 characters) that contains the needed security information.
- *'blanks'*:
 - When specified for the user ID and security is to be sent to the remote system, APPC uses the user's profile on the local system to send in the security field. If security is not to be sent, no user ID is sent.
 - When specified for the PASSWORD and security is to be sent to the remote system, APPC sends only an indicator that the password was specified. The user's password is not retrieved and sent. If security is not to be sent, no password is sent.

When a Remote Program Initiates the Transaction

For program start requests received by the AS/400 system, one of the following conditions must be met:

- On the AS/400 system, a default user can be specified on the communications entry for a subsystem description, refer to the *CL Reference* for a discussion of the DFTUSR parameter on the Add Communications Entry (ADDCMNE) command. For example, by specifying *NONE on the SECURITY keyword, an evoke function sent from one AS/400 system to another AS/400 system would cause the receiving AS/400 system to attempt to find a communications entry in the subsystem handling the program start request that allows a default user. If one is found, the receiving AS/400 system allows processing to continue. If one is not found, the program start request is rejected with a security violation error.
- If a user ID is provided without a password, then the user ID must have been verified by the remote system. SECURELOC(*YES) must also have been specified on the APPC configuration. The user ID is used to select an AS/400 user profile.
- A valid user ID and password are supplied.

Application Program Security for CPI Communications

The LU type 6.2 architecture defines various levels of security for the SECURITY parameter of the ALLOCATE and the MC_ALLOCATE verbs. The levels are SECURITY(NONE), SECURITY(SAME), and SECURITY(PGM). The CPI Communications architecture only defines the use of the first two values, NONE and SAME. The third value, PGM or program, specifies that the application program provides the security information. CPI Communications does not define an interface for specifying SECURITY(PGM). However, a CPI Communications program can be started using SECURITY(PGM) when the source program uses an LU 6.2 application programming interface that is not CPI Communications.

SECURITY(NONE) specifies that access security information is not to be sent when a request is sent to start a target program.

SECURITY(SAME) specifies that a calling user's ID is to be sent when a request is sent to start a target program; however, no password is sent with the request. Instead the user ID is sent with an indication that it has already been verified by the source system. AS/400 CPI Communications sends the user profile name that was used to start the source program as the already verified user ID.

An AS/400 CPI Communications program has no control over the security information that is sent. If the location initiating the program start request is secure, AS/400 CPI Communications will use SECURITY(SAME), and an already verified user ID will be sent on the program start request. If the location initiating the program start request is not secure, AS/400 CPI Communications will use SECURITY(NONE), and no security information will be sent on a program start request.

Using CPI Communications with SECURITY(NONE)

This section discusses how security is handled on a target AS/400 when SECURITY(NONE) is received.

Because no user ID accompanies a received program start request that specifies SECURITY(NONE), the default user profile that is configured in the communications entry in the subsystem description is used for received program start requests. If no default user profile is configured in the communications entry, program start requests will not be accepted.

Refer to the *Work Management Guide* for more information on defining communications entries.

Using CPI Communications with SECURITY(SAME)

CPI Communications programs that are started as a result of a received program start request that specified SECURITY(SAME) are started using the already verified user ID that accompanied the program start request.

Chapter 4. Running APPC

This chapter contains the information you need to run your network, including information on the Vary Configuration (VRYCFG) command and on the commands used to control modes.

Vary On and Vary Off Support

The Vary Configuration (VRYCFG) command is used to start and end the communications support.

Note: You can also start and end the communications support using the Work with Configuration Status (WRKCFGSTS) display. Refer to the *Communications Management Guide* for information about this support and additional information about the Vary Configuration command.

VRYCFG with STATUS(*ON) starts or activates the link between two or more systems and associates the communications support with a particular configuration consisting of line, controller, and device descriptions (if manually created).

VRYCFG with STATUS(*OFF) ends the link between two or more systems and releases the communications support and the configuration with which it is associated. When you specify VRYCFG with STATUS(*OFF), the association between the local and remote system is ended. No further communication is possible between the systems using the specified configurations.

The VRYCFG command has the following parameters:

CFGOBJ

Specifies the name of the description for the network interface, line, controller, or device to be varied on or off; or a list of names of configuration elements of the same configuration type, such as network interface, line, controller, or device type.

CFGTYPE

Specifies the type of configuration description to be varied on or off.

***NWI:** The network interface is varied.

***LIN:** The line is varied.

***CTL:** The controller is varied.

***DEV:** The device is varied.

STATUS

Specifies the status to vary the configuration object to.

***ON:** The object is varied on.

***OFF:** The object is varied off.

RANGE

Specifies what configuration elements should be varied, whether it is only the configuration element specified (*OBJ) or the configuration element specified and its attached configuration elements (*NET). For network interfaces, the attached configuration elements are lines, controllers, and devices. For lines, the attached configuration elements are controllers and devices. For controllers, the configuration elements are devices. Devices are considered not to have attached configuration elements. For devices there is no difference between specifying RANGE(*OBJ) or RANGE(*NET).

***NET:** All downline attached configuration elements are varied.

***OBJ:** Only the specified objects are varied.

VRYWAIT

Specifies whether the Ethernet, IDLC, tokenring, X.25, or switched SDLC line description is to be varied on asynchronously or synchronously. For a synchronous vary on operation, you can specify a wait time when an application will open or acquire a communications file immediately after issuing the vary on of the communications description.

***CFGOBJ:** Use the VRYWAIT parameter value specified in the line description.

***NOWAIT:** Do not wait for vary on completion. The line will vary on asynchronously.

WAIT-TIME: Specify a value from 15 to 180 seconds in 1-second intervals.

The system will wait until either the line is varied on before completing the VRYCFG command or until the timer expires.

ASCVRYOFF

Specifies if the configuration object (or objects if RANGE(*NET) is specified) is to be varied off synchronously or asynchronously. If the synchronous option is chosen, the configuration will be completely varied off before returning control to the user. Otherwise, the vary off request will fail. If the asynchronous option is chosen, control may be returned to the user before the vary off operation is completed.

***NO:** The vary off is done synchronously.

***YES:** The vary off is done asynchronously.

RESET

Specifies if the input/output processor (IOP) associated with the object is to be reset.

***NO:** The associated IOP is not reset.

***YES:** The associated IOP is reset.

Vary Configuration On Example

Using the network described in Figure 2-1 on page 2-4 and configured in “Configuration Example for a Two-System APPC Network (without APPN, Nonswitched)” on page D-3, the following example shows the VRYCFG commands that activate the line, controller, and devices on a non-switched line.

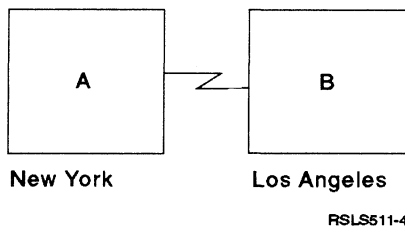


Figure 4-1. Two-System APPC Network (without APPN)

System A

```
VRYCFG CFGOBJ(LOSANGEL) CFGTYPE(*LIN) STATUS(*ON) RANGE(*NET)
```

System B

```
VRYCFG CFGOBJ(NEWYORK) CFGTYPE(*LIN) STATUS(*ON) RANGE(*NET)
```

Note: For programs communicating with each other on the same system, the controller description (*CTL) is the configuration type that should be specified.

Controlling Modes

This section contains information about controlling modes. Included is a description of the Start Mode (STRMOD) and End Mode (ENDMOD) commands, used to start and end modes with remote locations. Also described is the Change Session Maximum (CHGSSNMAX) command, used to control the number of sessions that are currently active between a local location and remote location using the specified mode.

Start Mode (STRMOD) Command

Note: Refer to “Using the Location Parameters” on page 3-4 for information on how the system processes the RMTLOCNAME, LCLLOCNAME, DEV, RMTNETID, and MODE parameters for this command.

The Start Mode (STRMOD) command starts a mode, enabling sessions to be established between the local location and remote location. The STRMOD command can be used to start one or all modes for an APPC configuration.

The STRMOD command is required only if the mode has been explicitly ended by a previous ENDMOD command. The APPC support issues an implicit STRMOD command when a device description is varied on.

The STRMOD command has the following parameters:

RMTLOCNAME

Specifies the remote location name. This parameter is required.

DEV

Specifies the device description name.

***LOC:** Specifies that the device description is to be determined by the system.

device name: Specify the name of the device description.

Note: The device description parameter is ignored if the system is using APPN support to communicate with the remote location specified as RMTLOCNAME. See “Using the Location Parameters” on page 3-4 for more details.

MODE

Specifies the mode that is to be started.

***NETATR:** Specifies that the mode specified in the network attributes is used.

***ALL:** Specifies that all configured modes for the specified remote location are to be started.

BLANK: A mode name consisting of 8 blank characters is used.

mode-name: Specify a mode name.

Note: SNASVCMG and CPSVCMG are reserved names and cannot be specified.

LCLLOCNAME

Specifies the name of your location.

***LOC:** Specifies that the local location name is to be determined by the system.

***NETATR:** Specifies that the default local location name specified in the network attributes is to be used.

local-location name: Specify the name of your location. The local location name is specified if you want to indicate a specific local location name for the remote location.

RMTNETID

Specifies the remote network ID used with the remote location.

***LOC:** Specifies that the system selects the remote network ID.

***NETATR:** Specifies that the remote network ID specified in the network attributes is used.

***NONE:** The remote network has no name.

remote-network-id: Specify the name of the remote network ID.

Example 1

```
STRMOD RMTLOCNAME(LOSANGEL) MODE(BLANK)
```

This command starts a mode named BLANK with a remote location named LOSANGEL. The device, local location name, and remote network identifier are selected by the system based on the remote location name LOSANGEL.

Example 2

```
STRMOD RMTLOCNAME(LOSANGEL) MODE(*ALL)  
LCLLOCNAME(NEWYORK) RMTNETID(APPN)
```

This command starts all configured modes that are currently not started. Because the device description uses the default, *LOC, the system selects the device description based on the remote location name LOSANGEL, the local location name NEWYORK, and the remote network identifier APPN.

End Mode (ENDMOD) Command

Note: Refer to "Using the Location Parameters" on page 3-4 for information on how the system processes the RMTLOCNAME, LCLLOCNAME, DEV, RMTNETID, and MODE parameters for this command.

The ENDMOD command ends one or more active modes. You can also specify how activities that have been requested on the remote system, but have not been performed are to be handled. This command is not required, but it can be explicitly issued whenever you wish to do so. Once an ENDMOD command is run, no sessions can be started between the local and remote locations on any mode that has ended until an explicit STRMOD command is run. However, a local session maximum of zero does not prevent a switched connection from being made. While the local session maximum is zero and a switched connection is made (either dial or answer), no communications occur on that mode until a STRMOD command is run to allow sessions to be established.

The ENDMOD command has the following parameters:

RMTLOCNAME

Specifies the remote location name for which one or more modes are to be ended. This parameter is required.

DEV

Specifies the device description name.

***LOC:** Specifies that the device description is to be determined by the system.

device-name: Specify a device description name.

MODE

Specifies the mode that is to be ended.

***NETATR:** Specifies that the mode specified in the network attributes is used.

***ALL:** Specifies all modes currently in use by the remote location are to be ended.

BLANK: The mode name, consisting of 8 blank characters, is to be used.

mode-name: Specify a mode name.

Note: SNASVCMG and CPSVCMG are reserved names and cannot be specified.

LCLLOCNAME

Specifies the name of your location.

***LOC:** Specifies that the local location name is to be determined by the system.

***NETATR:** Specifies that the local location name specified in the network attributes is used.

local-location-name: Specify the name of your location. The local location name is specified if you want to indicate a specific local location name for the remote location.

RMTNETID

Specifies the remote network ID used with the remote location.

***LOC:** Specifies that the system selects the remote network ID.

***NETATR:** Specifies that the remote network ID specified in the network attributes is used.

***NONE:** Specifies that remote network has no name.

remote-network-id: Specify the name of the remote network ID used.

CPLPDRQS

The complete pended requests parameter allows you to specify whether the remote location can complete work that is pending or if the work pending should be ended before being allowed to start.

***NO:** Specifies that requested activities currently in progress at the remote location can complete; activities that have been requested, but not started at the remote location are not performed.

***YES:** Specifies that all requested activities be allowed to complete before the mode is ended.

Example 1

```
ENDMOD RMTLOCNAME(LOSANGEL) MODE(BLANK)
```

This command ends mode named BLANK for remote location LOSANGEL. The device, local location name, and remote network ID are

selected by the system based on the remote location name LOSANGEL.

Example 2

```
ENDMOD RMTLOCNAME(LOSANGEL) MODE(*ALL)  
LCLLOCNAME(NEWYORK) RMTNETID(APPN) CPLPDRQS(*NO)
```

All currently active modes for remote location LOSANGEL are ended. Any work pending on the modes is not allowed to be completed. Because the device description was allowed to default to a value of *LOC, the system selects the device descriptions based on the remote location name LOSANGEL, the local location name NEWYORK, and the remote network identifier APPN.

Change Session Maximum (CHGSSNMAX) Command

Note: Refer to “Using the Location Parameters” on page 3-4 for information on how the system processes the RMTLOCNAME, LCLLOCNAME, DEV, RMTNETID, and MODE parameters for this command.

The Change Session Maximum (CHGSSNMAX) command is used to change the maximum number of sessions the local location allows a mode to have. When a change to the MAXSSN parameter is made, the remote location is informed and allowed to negotiate for a lower session maximum (the remote location cannot negotiate a session maximum higher than the value specified for the MAXSSN parameter). The session maximum value that results from the negotiation is called the *current session maximum*.

The following rules apply:

- Neither location may activate more sessions than the current session maximum.
- If the requested session maximum is accepted or negotiated by the remote location, the value requested on the CHGSSNMAX command is stored as the *local session maximum*.
- The remote location is not allowed to increase the current session maximum above the value stored as the local session maximum.

If a request to change the session maximum is rejected by the remote location, the CHGSSNMAX command ends abnormally and

the local session maximum is changed as follows:

- If the request was increasing the number of sessions, it is changed to the value specified on the MAXSSN parameter.
- If the request was decreasing the number of sessions, it is not changed.

This new value for the local session is only used the next time a new session maximum needs to be negotiated. The current session maximum, which controls how many sessions can be active between the local and remote location, is not changed if the command fails.

This command is normally used by the system operator to control the number of sessions that can be active at the same time with a remote location. This command is used only when the specified remote location and mode are active. If the current number of active sessions is greater than the maximum number specified on the command, no new sessions are created until the number of active sessions falls below that specified on the command. If the current number of active sessions is less than the maximum number specified, sessions may not be established until jobs requiring them are started.

The value determined by the locations remains in effect until another CHGSSNMAX command or an End Mode (ENDMOD) command is run for the same mode, or until all the device descriptions associated with the remote location are varied off.

Many CHGSSNMAX commands can be issued before the current maximum number of sessions ever become active. The number specified the last time the command was issued is the current *local session maximum* value.

If a vary off of the device description associated with the specified remote location is in progress, this command ends with an error.

Notes:

1. When this command is used to reduce the number of sessions with a remote location, the sessions that are ended first are the available locally controlled sessions, followed by any other available sessions. If the new session count is still not reached, other

sessions are ended as jobs using them are completed or are canceled.

2. When the CHGSSNMAX command is used to increase the maximum number of sessions that can be created with a remote location, the locally controlled sessions are made available first (depending on the negotiated values), and then other sessions are made available.
3. The CHGSSNMAX command does not change the value specified for the MAXSSN parameter in the mode description; the Change Mode Description (CHGMODD) command must be used to permanently change the value.

The CHGSSNMAX command has the following parameters:

RMTLOCNAME

Specifies the remote location name. This is a required parameter.

remote-location name: Specify the name of the remote location.

MAXSSN

Specifies the number of sessions allowed with the remote location. This value represents the desired maximum number of sessions for the specified mode name. It must be less than or equal to the MAXSSN parameter limit defined in the mode description. This value can be negotiated to a lower value by the remote location; therefore, the value specified here is not necessarily the value that is used.

Valid values for this parameter are 1 through 512.

DEV

Specifies the name of the device description to be used.

***LOC:** The device associated with the remote location is used. If several devices can be associated with the remote location, the system determines which device is used.

device-name: Specify the name of a device description that is associated with the remote location.

MODE

Specifies the name of the mode that is changed.

***NETATR:** Specifies that the mode name specified in the network attributes is used.

BLANK: A mode name (consisting of 8 blank characters) is used.

mode-name: Specify a value, no more than 8 characters, used to identify the mode that is changed.

Note: SNASVCMG and CPSVCMG are reserved names and cannot be specified.

LCLOCNAME

Specifies the local location name used.

***LOC:** Specifies that the local location name is to be determined by the system.

***NETATR:** The local location name that is in the network attributes is used.

local-location name: Specify the name of your location. Specify the local location if you want to indicate a specific local location name for the remote location.

RMTNETID

Specifies the remote network ID that is used with the remote location.

***LOC:** The system selects the remote network ID.

***NETATR:** The remote network ID specified in the network attributes is used.

***NONE:** The remote network has no name.

remote-network id: Specify a remote network ID.

Example

```
CHGSSNMAX RMTLOCNAME(APPCLC1) MODE(APPCCMOD1)
MAXSSN(10)
```

This command changes the maximum number of sessions allowed by the mode APPCCMOD1 for remote location APPCLC1 to a maximum of 10.

Displaying the Mode Status

You can use the Display Mode Status (DSPMODSTS) command to display the status of all mode entries for an APPC configuration. The display shows the following information:

- Mode name and status
- Device name and status
- Local location name

- Remote location name
- Additional information for conversations and sessions

This command is valid only for APPC device descriptions and if a mode is attached to the APPC device description.

The DSPMODSTS command has the following parameters:

DEV

Specifies the name of the APPC device description that contains the mode to be displayed.

MODE

Specifies the mode whose status is being displayed.

***ALL:** Specifies that all the modes used by the specified device are displayed.

mode: Specify the name (8 characters maximum) of the mode whose status is being displayed for the specified device.

BLANK: Specifies that the mode name of 8 blank characters is displayed.

OUTPUT

Specifies if the output from the command is shown at the requesting display station or printed with the job's spooled output on a printer.

*****: The output is shown (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

***PRINT:** The output is printed with the job's spooled output on a printer.

Examples of Displaying Mode Status

The following examples refer to the APPC network (without APPN) example described in Appendix D, in which a system at location NEWYORK (with a device description LOSANGEL) is communicating with a system at location LOSANGEL (with device description NEWYORK).

You can type the DSPMODSTS command and use the command prompt, or you can type the command with parameters on any command line and press the Enter key. The following is an example of the kind of display that is shown when you run the DSPMODSTS command and

specify LOSANGEL for the device and BLANK for the mode (that is, type DSPMODSTS DEV(LOSANGEL) MODE(BLANK) and press the Enter key):

```

DISPLAY DETAILS OF MODE STATUS
System: NEWYORK
Mode/status . . . . . : BLANK   Started
Device/status . . . . . : LOSANGEL ACTIVE
Local location/network ID . . . . . : NEWYORK APPC
Remote location/network ID . . . . . : LOSANGEL APPC

CONVERSATIONS:
TOTAL SOURCE TARGET DETACHED
Configured maximum . . . . . : 8
Number for device . . . . . : 1 1 0 0
Number for location . . . . . : 1 1 0 0

SESSIONS:
TOTAL LOCAL REMOTE
Configured limits . . . . . : 8 4
Local maximum . . . . . : 8
Negotiated limits . . . . . : 8 4 4
Number for device . . . . . : 1 1
Number for location . . . . . : 1 1

Press Enter to continue.
BOTTOM
F3=Exit F5=Refresh F12=Cancel F14=Display previous mode

```

This display shows the following information:

Mode/status

Name of the mode description and status of the mode. One of the following may be displayed for the status of the mode:

Ended

Indicates that the mode cannot be used for communications. The local system has issued an ENDMOD command to end the mode. Only the local system can start the mode.

Reset

Indicates that the mode cannot be used for communications. Either the mode is in the initial state (not yet started), or it has been ended by the remote system. Either the local or the remote system may attempt to start the mode.

Started

Indicates that the mode has been started and can be used for communications.

Unknown

Status of mode could not be determined.

Device/status

Name of the APPC device and status of the APPC device. One of the following may be displayed for the status of the device:

Varied off pending

The device is in the process of being varied off.

Varied off

The device is not being used for communications.

Vary on pending

The device is in the process of being varied on.

Varied on

The device is varied on.

Active

The device is ready to handle APPC sessions.

Held

The user or the system held the communications device to prevent it from participating in communications.

RCYPND

Error recovery is pending for the device.

RCYCNL

Error recovery was canceled for the device.

Failed

This status indicates that an error occurred for the device that can only be recovered by varying the device off and on again.

***Damaged**

The device object has received hard damage.

***Locked**

The status of the device could not be determined because another job had an exclusive lock on the device.

***Unknown**

The status indicator of the device cannot be determined.

Local location/network ID

Name of the local location and the network ID associated with the local location. The name and network ID can each have up to 8 characters.

Remote location/network ID

Name of the remote location and the network ID associated with the remote location. The name and network ID can each have up to 8 characters.

Conversations

Configured maximum

How many were configured for this device.

Number for device

Total

Current number running on this device.

Source

Current number of conversations that are allocated to source programs on this device.

Target

Current number of conversations that are allocated to target programs on this device (started as a result of a received program start request).

Detached

Current number of conversations that are not active on a session but have not been detached from the program on this device.

Number for location

Total

Current number running for this location.

Source

Current number of conversations that are allocated to source programs for this location.

Target

Current number of conversations that are allocated to target programs for this location (started as a result of a received program start request).

Detached

Current number of conversations that are not active on a session but have not been detached from the program for this location.

Sessions

Configured limits

Total

How many were configured for this device.

Local

Number of locally controlled sessions configured.

Local maximum

Maximum number of sessions requested by the most current CHGSSNMAX command.

Negotiated limits

Total

Current maximum number of sessions allowed for this device.

Local

Current minimum number of locally controlled sessions on this device.

Remote

Current minimum number of remotely controlled sessions on this device.

Number for device

Total

Current active number of sessions on this device.

Local

Current active number of locally controlled sessions on this device.

Number for location

Total

Current active number of sessions running for this location.

Local

Current active number of locally controlled sessions for this location.

To display all modes for a device, you can type DSPMODSTS DEV(LOSANGEL) MODE(*ALL) and then press the Enter key to obtain the following display:

```

                                DISPLAY MODE STATUS
                                System:  NEWYORK
Device . . . . . : LOSANGEL
Device status . . . . . : ACTIVE

Type options, press Enter.
S=Display details

OPT  MODE  STATUS  TOTAL  SOURCE  TARGET  DETACHED
-   BLANK  Started   0      0      0      0
-   #BATCH Started   0      0      0      0
-   SNASVCMG Started  0      0      0      0

```

If you then press F11 (Display sessions), the following display is shown:

```

      DISPLAY MODE STATUS
      System: NEWYORK
Device . . . . . : LOSANGEL
Device status . . . . . : ACTIVE

Type options, press Enter.
5=Display details

OPT  MODE      MODE      --SESSIONS--
    MODE  STATUS  TOTAL  LOCAL
- BLANK  Started  0      0
- #BATCH Started  0      0
- SNASVCMG Started  0      0

```

These displays show the device name and status. Current values are also shown for conversations and sessions. From these displays you may also select option 5 (Display details) for a selected mode.

The following examples depict how the mode controlling commands (STRMOD, ENDMOD, and CHGSSNMAX) change the affected modes. The DSPMODSTS command is used throughout this example to show the transitions that are taking place for the modes affected by these commands. This example shows the mode controlling commands being run at system NEWYORK as they relate to the connection with location LOSANGEL.

To show the following display, type DSPMODSTS DEV(LOSANGEL) MODE(*ALL) and press F11 (Display sessions) to display information about the sessions:

```

      DISPLAY MODE STATUS
      System: NEWYORK
Device . . . . . : LOSANGEL
Device status . . . . . : ACTIVE

Type options, press Enter.
5=Display details

OPT  MODE      MODE      --SESSIONS--
    MODE  STATUS  TOTAL  LOCAL
- BLANK  Started  1      1
- #BATCH Started  1      1
- SNASVCMG Started  1      1

```

the modes after the connection between NEWYORK and LOSANGEL has been established. The display shows that the reserved mode SNASVCMG and that the modes #BATCH and BLANK have been started. The SNASVCMG mode is used by the system for negotiating SNA sessions between two locations as well as alerts between network nodes. (In SNA, an **alert** is a record sent to a focal point to identify a problem or an impending problem.)

The following display shows what happens when the local system has started an application that

acquires a session. The mode specified on the DSPMODSTS command is #BATCH (Note that this is one of the modes specified on the CRTDEVAPPC command when NEWYORK and LOSANGEL were configured. Refer to Appendix D for details).

```

      DISPLAY MODE STATUS
      System: NEWYORK
Device . . . . . : LOSANGEL
Device status . . . . . : ACTIVE

Type options, press Enter.
5=Display details

OPT  MODE      MODE      -----CONVERSATIONS-----
    MODE  STATUS  TOTAL  SOURCE  TARGET  DETACHED
- BLANK  Started  0      0      0      0
- #BATCH Started  1      1      0      0
- SNASVCMG Started  0      0      0      0

```

Pressing F11 (Display sessions) gives the previous display showing display mode status for sessions. The displays also show that there is one session that has been established and one conversation currently in progress between NEWYORK and LOSANGEL using the mode #BATCH.

Because all modes for APPN(*NO) start automatically when the device description is varied on, you must first end a mode using the ENDMOD command before you can issue a STRMOD command to start a mode. You can then start an IBM-supplied mode BLANK to the remote location LOSANGEL by issuing the STRMOD command:

```
STRMOD RMTLOCNAME(LOSANGEL) MODE(BLANK)
```

Then press the Enter key to start the mode. (Note that the mode BLANK is the other mode specified on the CRTDEVAPPC command during configuration).

By next issuing the DSPMODSTS command DSPMODSTS DEV(LOSANGEL) MODE(BLANK)

you will see that the mode BLANK has been started, and also the session limits that have been negotiated between NEWYORK and LOSANGEL. Because the DSPMODSTS command was started by supplying the mode name, the display that is shown is the one that shows the details of the mode status:

```

                DISPLAY DETAILS OF MODE STATUS
                                     System: NEWYORK
Mode/status . . . . . : BLANK      Started
Device/status . . . . . : LOSANGEL  ACTIVE
Local location/network ID . . . . . : NEWYORK  APPC
Remote location/network ID . . . . . : LOSANGEL  APPC

CONVERSATIONS:
Configured maximum . . . . . : 8
Number for device . . . . . : 0      0      0      0
Number for location . . . . . : 0      0      0      0

SESSIONS:
Configured limits . . . . . : 8      4
Local maximum . . . . . : 8
Negotiated limits . . . . . : 8      4      4
Number for device . . . . . : 0      0
Number for location . . . . . : 0      0

                                BOTTOM
Press Enter to continue.

F3=Exit  F5=Refresh  F12=Cancel  F14=Display previous mode

```

```

                DISPLAY DETAILS OF MODE STATUS
                                     System: NEWYORK
Mode/status . . . . . : BLANK      Started
Device/status . . . . . : LOSANGEL  ACTIVE
Local location/network ID . . . . . : NEWYORK  APPC
Remote location/network ID . . . . . : LOSANGEL  APPC

CONVERSATIONS:
Configured maximum . . . . . : 8
Number for device . . . . . : 0      0      0      0
Number for location . . . . . : 0      0      0      0

SESSIONS:
Configured limits . . . . . : 8      4
Local maximum . . . . . : 6
Negotiated limits . . . . . : 6      3      3
Number for device . . . . . : 0      0
Number for location . . . . . : 0      0

                                BOTTOM
Press Enter to continue.

F3=Exit  F5=Refresh  F12=Cancel  F14=Display previous mode

```

To change the maximum number of sessions allowed between locations NEWYORK and LOSANGEL, you can use the CHGSSNMAX command. If you want to change the maximum number of sessions specified for the mode BLANK from 8 to 6, for example, type

```
CHGSSNMAX RMTLOCNAME(LOSANGEL) MODE(BLANK)
MAXSSN(6)
```

To show the mode status, type
 DSPMODSTS DEV(LOSANGEL) MODE(*ALL)

As this display shows, the configured limits for the mode are not affected by the CHGSSNMAX command. What has changed are the negotiated limits between NEWYORK and LOSANGEL using mode BLANK.

To end all of the user modes between NEWYORK and LOSANGEL, you can use the ENDMOD command. Once this command has been run, no new sessions can be started between the two locations on modes BLANK and #BATCH until another explicit STRMOD command is run. Type

```
ENDMOD RMTLOCNAME(LOSANGEL) MODE(*ALL)
```

To display the mode status, type
 DSPMODSTS DEV(LOSANGEL)

```

                DISPLAY MODE STATUS
                                     System: NEWYORK
Device . . . . . : LOSANGEL
Device status . . . . . : ACTIVE

Type options, press Enter.
5=Display details

OPT  MODE  MODE  STATUS  TOTAL  SOURCE  TARGET  DETACHED
5  BLANK  Started  0      0      0      0
-  #BATCH Started  1      1      0      0
-  SNASVCMG Started  0      0      0      0

```

Use Option 5 to display the details of the mode status for the mode BLANK:

```

                DISPLAY MODE STATUS
                                     System: NEWYORK
Device . . . . . : LOSANGEL
Device status . . . . . : ACTIVE

Type options, press Enter.
5=Display details

OPT  MODE  MODE  STATUS  TOTAL  SOURCE  TARGET  DETACHED
5  BLANK  Ended  0      0      0      0
5  #BATCH Ended  1      1      0      0
-  SNASVCMG Started  0      0      0      0

```

Note: Only the user modes #BATCH and BLANK are affected by the ENDMOD command. The reserved mode SNASVCMG is not affected by the ENDMOD command.

Use Option 5 to display the details of the mode status for the mode #BATCH:


```

DISPLAY DETAILS OF MODE STATUS
System: NEWYORK
Mode/status . . . . . : #BATCH      Ended
Device/status . . . . . : LOSANGEL   ACTIVE
Local location/network ID . . . . . : NEWYORK   APPC
Remote location/network ID . . . . . : LOSANGEL   APPC

CONVERSATIONS:
Configured maximum . . . . . :      0
Number for device . . . . . :      1      1      0      0
Number for location . . . . . :      1      1      0      0

SESSIONS:
Configured limits . . . . . :      0      4
Local maximum . . . . . :      0
Negotiated limits . . . . . :      0      0      0
Number for device . . . . . :      1      1
Number for location . . . . . :      1      1

Press Enter to continue.
BOTTOM
F3=Exit  F5=Refresh  F12=Cancel  F14=Display previous mode

```

The ENDMOD command causes the mode #BATCH to end, but it does not affect sessions that have already been activated. This is the reason that there is still one active session and one active conversation between NEWYORK and LOSANGEL using mode #BATCH.

To display the mode status, type
 DSPMODSTS DEV(LOSANGEL)

The DSPMODSTS command shown here occurred after the application that had acquired a session using mode #BATCH has ended. This caused the session and conversation counts to go to zero.

```

DISPLAY MODE STATUS
System: NEWYORK
Device . . . . . : LOSANGEL
Device status . . . . . : ACTIVE

Type options, press Enter.
5=Display details

-----CONVERSATIONS-----
OPT  MODE      STATUS      TOTAL  SOURCE  TARGET  DETACHED
-    BLANK     Ended       0      0      0      0
-    #BATCH    Ended       0      0      0      0
-    SNASVCMG Started     0      0      0      0

```

Chapter 5. Writing ICF APPC Application Programs

This chapter describes how an application program uses the AS/400 system, intersystem communications function (ICF) file, and APPC to communicate with a remote system. The program can be coded using any of the high-level languages that support ICF, such as C/400, COBOL/400, FORTRAN/400 and RPG/400. These languages allow an application program to do the following:

- Start a session by opening a file and acquiring a program device either explicitly or implicitly.
- Send or receive information by writing to or reading from a program device.
- End a session by releasing the program device and closing the file.

Note: FORTRAN/400 supports only the following operations:

- Open with acquire
- Close
- Read
- Write

For more information about the languages, refer to the appropriate language reference manual. Chapter 7 contains additional information about writing applications that use APPC.

This chapter also contains the following information:

- A description of the read and write operations that specify a record format, which contains specific communications functions. You can define record formats using data description specifications (DDS) keywords, or you can use system-supplied formats.
- Information about using return codes. After an ICF operation completes, a return code (and a high-level language file status) is returned to your application. The return code indicates whether the operation completed successfully or unsuccessfully. Along with the return code, exception messages may also be issued. Refer to Appendix B for more information about return codes and to the appropriate language reference manuals for more information about the high-level language file status.

- A mapping of the LU type 6.2 architected verbs and parameters to the corresponding ICF operation or function. This information is useful if you are writing application programs that are used for communications between an AS/400 system and another system, such as a System/370 or personal computer, that supports APPC.

Intersystem Communications Function File

An intersystem communications function (ICF) file must be created before your application can use APPC. For more information about the ICF file, see the *ICF Programmer's Guide*.

The ICF file is a system object of type *FILE with a specific set of commands and operations. The commands allow you to manage the attributes of the file and the operations allow a program to use the file. Commands allow you to create, delete, change and display the file description.

The following commands are valid for APPC applications.

CR TICFF	Create ICF file and file level attributes. This command allows you to create an ICF file.
CHGICFF	Change ICF file. This command allows you to change the file attributes of the ICF file.
OVRICFF	Override ICF file. This command allows you to temporarily change the file attributes of the ICF file at run time. These changes are only in effect for the duration of the job and do not affect other users of the file.
DLTF	Delete file. This command allows you to delete a file from the system.
DSPFD	Display file description. This command displays the file description of any file on the

system. The information may be printed or displayed.

DSPFFD

Display file field description. This command displays the description of the fields in any file on the system. This information may be printed or displayed.

ADDICFDEVE

Add ICF device entry. This command allows you to permanently add a program device entry that contains a program device name, remote location information, and session level attributes.

CHGICFDEVE

Change ICF device entry. This command allows you to permanently change the program device attributes previously added with the ADDICFDEVE command.

OVRICFDEVE

Override ICF device entry. This command allows you to:

- Temporarily add the program device entry, the remote location information, and the session level attributes to the ICF file.
- Override a program device entry with the specified remote location information and session level attributes for an ICF file.

RMVICFDEVE

Remove ICF device entry. This command allows you to permanently remove the program device entry previously added with the ADDICFDEVE command or changed with the CHGICFDEVE command.

(RMTLOCNAME, DEV, LCLLOCNAME, RMTNETID, and MODE), refer to “Using the Location Parameters” on page 3-4.

PGMDEV

Specifies the program device name that is defined in the ICF file and specified in the application. The total number of devices that may be acquired to an ICF file is determined by the MAXPGMDEV parameter on the CRTICFF or CHGICFF command.

pgm-device name: Enter the name by which the user program refers to this communications session.

RMTLOCNAME

Specifies the remote location name with which your program communicates. A remote location must be specified using the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. If a remote location name is not specified, a major and minor error return code is returned when an attempt is made to acquire the program device.

***REQUESTER:** The name used to refer to the communications device through which the program was remotely started.

location-name: Enter the name of the remote location that should be associated with the program device.

DEV

Specifies the name of the device description used for the remote location. If the device is not valid for the remote location, a major and minor error return code is returned when an attempt is made to acquire the program device.

***LOC:** Specifies that the device is to be determined by the system.

device-name: Specify the name of the device that is associated with the remote location.

LCLLOCNAME

Specifies the name of your location. If the local location name is not valid for the remote location, a major and minor error return code is returned when an attempt is made to acquire the program device.

***LOC:** Specifies that the local location name is to be determined by the system.

Specifying the Program Device Entry Commands

The following describes the parameters for the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands and lists the values for each parameter for APPC. For more information about how the system processes the location parameters

***NETATR:** Specifies that the local location name specified in the system network attributes is used.

local-location name: Specify the name of your location.

MODE

Specifies the mode to be used.

***NETATR:** Specifies that the mode specified in the system network attributes is used.

BLANK: The mode name, consisting of 8 blank characters, is used.

mode-name: Specify a mode name for the remote location. If the mode is not valid for any combination of remote location device, local location, and remote network ID, a major and minor error return code is returned when an attempt is made to acquire the program device.

RMTNETID

Specifies that the remote network ID used with the remote location.

***LOC:** Specifies that the remote network ID for the remote location should be used. If several remote network IDs are associated with the remote location, the system automatically selects the remote network ID.

***NETATR:** Specifies the remote network ID specified in the network attributes is used.

***NONE:** The remote network has no name.

remote-network id: Specify a remote network ID.

FMTSLT

Specifies the type of record selection used for input operations.

***PGM:** The program determines what record formats are selected.

***RECID:** The RECID keywords specified in the DDS for the ICF file are used to select records.

***RMTFMT:** The remote format name received from the sending system is used to select records.

CNVTYPE

Specifies the conversation type for which the application program is designed. For additional information, refer to "APPC Sessions and Conversations" on page 3-1.

***SYS:** Specifies that the system gives the length and general data stream identifier values that go before each section of user data in APPC communications. The application gives the data portion of the general data stream on output operations and receives only the data portion of the general data stream on input operations. For the LU type 6.2 architecture, this is the mapped conversation support.

***USER:** Specifies that the application program gives the length and general data stream identifier values that go before each section of user data. The application gives the length and general data stream values in the first 4 bytes of output data and receives the length and general data stream values in the first 4 bytes of input data. For the LU type 6.2 architecture, this is the basic conversation support.

***SRCPGM:** Specifies that the target program accepts the conversation type specified by the source program.

OVRFLWDTA

Specifies whether overflow data is discarded or retained. Overflow data is the excess data that occurs when your print buffer is not able to receive all of a lengthy input operation.

***DISCARD:** Specifies that overflow data is truncated.

***RETAIN:** Specifies that overflow data is kept and can be obtained on the next input operation.

Communications Operations and Functions

This section gives a brief description of the operations and functions you can code into a program that uses the APPC support to communicate with another system.

Common functions such as RECID are not described in this publication. You should refer to the *ICF Programmer's Guide* or the *DDS Reference* for additional information.

Starting a Session Using the Open and Acquire Operations

A communications **session** is a logical connection between two systems through which a local program can communicate with a program at a remote location. Your application program uses the **open** and **acquire** operation to establish a session, which is controlled by your system (locally controlled) or by the remote system (remotely controlled).

Starting a Transaction Using the Evoke Function

A **transaction** is a logical connection between two programs, and is equivalent to establishing a conversation between the two programs. Your program uses the **evoke** function to start a program on the remote system after a session is started. The evoke function causes a program start request (an APPC FMH5) to be sent. It is only valid when your program is not already communicating with another program on the same session.

You can use the **defer evoke** (DFREVOKE) keyword with the EVOKE keyword to delay the evoke function until either the send buffer is full, or an operation using an ALWRT, CONFIRM, DETACH, FRCDTA, or INVITE keyword is issued. The DFREVOKE keyword is useful for specialized applications in which data must be sent at the same time as the program start request.

Note: If you have an APPC device configured with the single session location as *YES (that is, SNGSSN(*YES)), you can only have one active session and one active transaction at any given time. However, if your program acquires a session, uses an evoke function to start a program on that session, and then sends or receives a detach function, your program is no longer connected to the session it previously acquired. A program running in a different job can then acquire your old session, provided a conversation is available on the mode associated with that session. If your program attempts another evoke function to start a program on the old session, it will fail.

With the evoke function your program can specify the following information:

- Remote program name

- Remote library name (optional)
- User-defined program initialization parameters (optional)
- Synchronization level (optional)
- Security information (optional)

If your program uses the EVOKE, SECURITY, and SYNVLV keywords, you can specify all of the above information. If your program is using one of the evoke system-supplied formats, you can specify all of the above except synchronization level. In this case the default for the synchronization level is always *NONE.

For information on how to code the evoke function refer to the *ICF Programmer's Guide* and the *DDS Reference*.

Notes:

1. If a program start request is received on the AS/400 system with an unqualified program name, (that is, it has no library name) the system uses the library list specified on the QUSRLIBL system value at the time the subsystem that is handling the program start request was started.
2. If a program start request is received on the AS/400 system with a qualified program name, the program name can be in the form 'program.library' or in the form 'library/program'.
3. Program names and library names on the AS/400 system are limited to 10 characters each.

Program Initialization Parameters:

When using EVOKE DDS keyword

If your program is using DDS keywords, it may send up to 255 user-defined parameters to the remote system. The number and format of the parameters is defined by the target program. If the remote system is another AS/400 system, the parameters are passed to the target program as if they were passed from a call to a program (CALL) command. For a prestart job program, the parameters are retrieved from a data area using the Retrieve Data Area (RTVDTAARA) command or the appropriate high-level language oper-

ation (such as the COBOL ACCEPT statement).

If the remote system is an AS/400 system, a maximum of 40 parameters may be passed. The total length of the user-defined parameters cannot exceed 32,767 characters. For a pre-start job, the length of the user-defined parameters cannot exceed 2000 bytes. To determine the total length of the data to be sent, use the following formula:

$$4 + (4 + \text{length}(\text{parm1}) + 4 + \text{length}(\text{parm2}) + \dots + 4 + \text{length}(\text{parmn}))$$

The constant 4 must be included to allow for system overhead.

When using evoke system-supplied formats

If your program is using one of the evoke system-supplied formats, you specify the parameters in the user buffer. Because this does not allow you to specify individual parameters, you can only pass one parameter or you must know how the remote system separates program initialization parameters and use that separator to code the parameters in the user buffer. For example, a System/36 separates parameters with a comma. If you want the program initialization parameter data to be treated as individual parameters by a System/36, you should code your user buffer with each parameter separated by a comma.

Notes:

1. When sending program parameters as part of the evoke data stream, ensure each parameter sent is the same length as the respective parameter in the target program. If it is longer than the respective target program parameter, truncation occurs. If it is shorter, unpredictable results may occur.
2. For program start requests received on an AS/400 system, commas embedded within parameters cause what would otherwise be a single parameter to be considered multiple parameters. For example:

```
Parm 1="A"  
Parm 2="B,C"  
Parm 3="D,E"
```

Instead of being three parameters, this example is considered five separate parameters.

3. The DFREVOKE keyword is not supported by system-supplied formats.

These notes pertain to both the DDS keyword and the system-supplied formats for specifying program initialization parameters.

Synchronization Level: When using DDS keywords, you specify the synchronization level by using the SYNLVL keyword. The SYNLVL keyword supports the following values:

- *NONE: Specifies that confirm processing is not allowed on this transaction. This is the default value.
- *CONFIRM: Specifies that the sending program can request the receiving program to acknowledge receipt of the data. The receiving program can send a positive acknowledgement, or the receiving program or system can send a negative acknowledgement. Refer to "Confirm Considerations" on page 7-3 for additional information concerning confirm processing. Also, refer to the descriptions of the CONFIRM, RSPCONFIRM, and RCVCONFIRM keywords for additional information on confirm processing.

Notes:

1. Your program cannot specify a synchronization level when using one of the evoke system-supplied formats. In this case the synchronization level always defaults to *NONE.
2. Simply specifying SYNLVL(*CONFIRM) does not cause confirm processing to occur; it only means that confirm processing will be allowed. To perform confirm processing, you must specify the CONFIRM keyword.

Security: If your program is using DDS keywords, security information for the evoke function is provided with the SECURITY keyword. If your program is using one of the evoke system-supplied formats, security information is provided in the evoke parameter list coded in your program. The default value is to send no security information with the evoke function.

Refer to “APPC Security Considerations” on page 3-6 for information about APPC security.

Sending Data

You can send data during a transaction using the **write** operation. APPC also supports various functions that are discussed below. These functions may be issued by your program to another program with or without data.

Note: These functions can be used only when your program is in send state. For more information about sending and receiving data, see “Conversations” on page 3-1.

Control-Data Function

Your program uses the **control-data** (CTLDTA) keyword at the file or record level to inform the remote program that control data is being sent. The CTLDTA keyword signals program-specific data that is not considered normal data flow.

The CTLDTA keyword has no additional affect when the EOS, RSPCONFIRM, or RQSWRT keywords are in effect.

Force-Data Function

Your program uses the **force-data** (FRCDTA) function to immediately send communications data currently held in the buffer without waiting for the buffer to become full. Your program can continue to send data without waiting for confirmation to be returned, but your program must be in send state.

The FRCDTA keyword has no additional effect when the keywords ALWWRT, CONFIRM, DETACH, EVOKE, INVITE, or FAIL are also selected.

Confirm Function

Your program uses the **confirm** function to indicate the end of a user-defined group of data and requests that the remote program acknowledge that it has accepted the data. An operation that includes the confirm function does not complete until the remote program responds with a positive or negative acknowledgment.

Note: The confirm function only applies when SYNLVL(*CONFIRM) is specified in the EVOKE

DDS record format used by the source program, or when the program start request received by a target program establishes a synchronization level of confirm. An AS/400 target program can determine the synchronization level established by the source program by using the get-attributes operation. Refer to “Get-Attributes Operation” on page 5-10 for more information.

The confirm function causes any data currently held in the buffer to be sent, including any data on a write operation that specified the confirm function. Refer to “Confirm Considerations” on page 7-3 for additional information.

Format-Name Function

Your program uses the **format-name** (FMTNAME) function to send the record format name, along with any data specified, to the remote system. The format name can only be used with mapped conversations (conversations in which CNVTYPE (*SYS) is specified on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands for the source program. This function should also only be used if the remote system is able to receive the APPC architected map name GDS ID variable. For example, the AS/400 system and System/38 support receipt of map names; System/36 does not.

When sending and receiving format names between AS/400 systems, you should specify FMTSLT(*RMTFMT) on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE commands.

Variable-Buffer-Management Function

If you want to send multiple or partial records using one write operation, then your program can use the variable-buffer-management (VARBUFMGMT) function.

Using the VARBUFMGMT keyword allows you to specify the length of data independently of the data itself. A program uses the data length specified as the value passed in the variable length (VARLEN) DDS keyword, or if VARLEN is not used, the length of the record format specified on the write operation is used. Note that the length specified on the VARLEN keyword must be greater than zero if it is used.

Note: The variable-buffer-management function can only be used with basic conversations (that

is, with CNVTYPE (*USER) on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE commands for the source program, or with CNVTYPE (*USER) or CNVTYPE (*SRCPGM) for the target program).

Figure 5-1 is a COBOL/400 example program for sending fixed-length records. This example shows only one of many ways to perform this task. You can also write a program that will send multiple variable-length records. In addition, other programming languages may be used.

```

R EXAMPLE1          VARBUFMT
REC1              30

R EXAMPLE2          VARBUFMT
                  VARLEN(&VLEN)
REC2              30
VLEN              5S P

```

Figure 5-1. DDS Used for Variable-Buffer-Management Example Program

```

01 VARBUF1.
   05 ARRAY OCCURS 3 TIMES.
      10 RECORD-LENGTH PIC 99 COMP-4.
      10 RECORD-DATA PIC X(8).

SEND-FULL-FORMAT.
*****
* The next WRITE will be done to a record format without the *
* VARLEN keyword specified; this will cause the length of the *
* entire record format to be used. Three ten-character records *
* will be sent.
* Note: This format also uses the VARBUFMT keyword to remove *
* the limitation of exactly one logical record.
*****

MOVE 10 TO RECORD-LENGTH(1).
MOVE "RECORD 1" TO RECORD-DATA(1).
MOVE 10 TO RECORD-LENGTH(2).
MOVE "RECORD 2" TO RECORD-DATA(2).
MOVE 10 TO RECORD-LENGTH(3).
MOVE "RECORD 3" TO RECORD-DATA(3).
WRITE ICFREC FROM VARBUF1 FORMAT IS "EXAMPLE1"
   TERMINAL IS PGM-DEV-NME.

SEND-PARTIAL-FORMAT.
*****
* The next WRITE will be done to a record format with the *
* VARLEN keyword specified; this will allow less than the *
* entire record format to be used. Two ten-character records *
* will be sent using a record format.
* Note: This format also uses the VARBUFMT keyword to remove *
* the limitation of exactly one logical record.
*****

MOVE 10 TO RECORD-LENGTH(1).
MOVE "RECORD 4" TO RECORD-DATA(1).
MOVE 10 TO RECORD-LENGTH(2).
MOVE "RECORD 5" TO RECORD-DATA(2).

MOVE VARBUF1 TO REC2 OF EXAMPLE2-0.
MOVE 20 TO VLEN OF EXAMPLE2-0.
WRITE ICFREC FORMAT IS "EXAMPLE2"
   TERMINAL IS PGM-DEV-NME.

```

Figure 5-2. COBOL/400 Example Program for Sending Multiple Fixed-Length Records

Receiving Data

Your application program uses the **read** operation to obtain data or control information from the remote system. The read operation is valid while the transaction is active and is in either the receive or send state. Using the read operation in send state causes your program to send the change-direction indicator to your partner program, after which your program enters receive state.

If you do not use the variable-buffer-management (VARBUFMT) function, and you request less data than the length of the record you are receiving, the remaining data to be received is handled according to the OVRFLWDTA parameter on the ICF device entry being used.

Invite Function

Your application program uses the **invite** function to request input data from a remote program, but it receives control without waiting for the input. To obtain the data, your program must issue a read-from-invited-program-devices or read operation during the transaction. The invite function is valid while the transaction is active and in either the receive or send state. Using the invite function while in send state causes your program to send the change-direction indicator to your partner program, after which your program enters the receive state.

Read-from-Invited-Program-Devices Operation

Your program can use the **read-from-invited-program-devices** operation to obtain data from any program that has responded to an invite function that was previously issued in your program. If data becomes available to your program from more than one program device before the read-from-invited-program-devices operation is issued, your program receives the data that was *first* made available.

Using the Variable-Buffer-Management Function on Read Operations

If you want to receive multiple or partial records instead of just one record, using one read operation, then your program can use the variable-buffer-management (VARBUFMGT) function.

You should also note that basic conversations using the VARBUFMGT function differ from basic conversations that do not use the function and from mapped conversations in the following manner:

- If you use VARBUFMGT, and you request less data than the length of the record you are receiving, the remaining data to be received is saved until your next read operation.
- If you do not use VARBUFMGT, and you request less data than the length of the record you are receiving, the remaining data to be received is handled according to the OVRFLWDTA parameter on the ICF device entry being used.

Figure 5-3 is a COBOL/400 example program for receiving fixed-length records. This example shows only one of many ways to perform this task. You can also write a program that will receive multiple variable-length records. In addition, other programming languages may be used. Note that the DDS used for this example is the same as the DDS used in the example program for sending fixed-length records in Figure 5-1 on page 5-7.

```

01 VARBUF1.
05 ARRAY OCCURS 3 TIMES.
10 RECORD-LENGTH PIC 99 COMP-4.
10 RECORD-DATA PIC X(8).

RECEIVE-RECORDS.
*****
* The following READ will receive 0,1,2, or 3 records (10
* bytes per record) and display the data from each record.
* Note: this format uses the VARBUFMGT keyword to remove
* the limitation of exactly one logical record.
*****

READ ICFFILE INTO VARBUF1
  FORMAT IS "EXAMPLE1"
  INDICATORS ARE ICFF-INDIC-AREA.
ACCEPT IO-FBA FROM IO-FEEDBACK FOR ICFFILE.
DISPLAY ACTUAL-RECEIVE-LENGTH.
IF ACTUAL-RECEIVE-LENGTH < 1 THEN
  DISPLAY "NO DATA RECEIVED"
ELSE
  DISPLAY RECORD-DATA(1).
  IF ACTUAL-RECEIVE-LENGTH > 10 THEN
    DISPLAY RECORD-DATA(2).
  IF ACTUAL-RECEIVE-LENGTH > 20 THEN
    DISPLAY RECORD-DATA(3).

```

Figure 5-3. COBOL/400 Example Program for Receiving Multiple Fixed-Length Records

Waiting for a Display File, ICF File, or Data Queue

A data queue is an object that is used to communicate and store data used by several programs in a job or between jobs. You can use **data queues** when your program waits for one or more of the following to occur:

- Pressing an enabled command key or Enter key from an invited display device
- Data becoming available when the session is invited for an ICF program device
- A user-defined entry being made to a data queue by a job running on the system

To indicate that a data queue will have entries placed in it when your program has to wait for any of the three actions listed above, use one of the following commands with the specified data queue (DTAQ) parameter:

```

Create Display File (CRTDSPF)
Change Display File (CHGDSPF)
Override Display File (OVRDSPF)
Create ICF File (CRTICFF)
Change ICF File (CHGICFF)
Override ICF File (OVRICFF)

```

Refer to the *ICF Programmer's Guide* for more information.

Notifying the Remote Program of Problems by Using the Fail Function

Your program uses the **fail** function to indicate that it has detected an abnormal condition while it was sending or receiving data. If a program is in a receive state, the fail function causes a negative-response indication and an APPC FMH7 to be sent to the remote system. If the program is in a send state, the fail function causes an APPC FMH7 to be sent to the remote system.

No data can be sent with a fail function. The fail function is also used to send a negative response to a received confirm request.

If a program that is in the send state issues a fail function, it may indicate that the data just sent was in error or that some other condition occurred. All records sent before the fail function was issued are still sent to the remote program.

If a program that is in the receive state issues a fail function, it indicates that the data received was in error. The program that sent the fail function should immediately do at least one output operation so it can indicate why it sent the fail operation. The record sent by the output operation should identify what the error is and where the remote program should start again. Data received by APPC, but not yet given to the application program, is lost after a fail function has been issued by a program in receive state.

The program that issued the fail function will be in a send state, and the program that receives the fail function will be in a receive state. This allows the program that was sending data to determine which record failed or which record it should begin sending again.

If both programs issue a fail function at the same time, the program that was receiving will be successful and should send. The program that was sending will receive a fail return code.

Using Additional Functions and Operations

APPC supports the additional functions and operation listed below. Refer to the *ICF Programmer's Guide* for more information on these functions and operation.

Respond-to-Confirm Function

Your program uses the **respond-to-confirm** (RSPCONFIRM) function to send a positive response to a received confirm request. The RSPCONFIRM function can only be used when a confirm request is outstanding. You can check the major and minor return codes or use the receive confirm (RCVCONFIRM) indicator to determine when to issue a RSPCONFIRM function. After issuing the RSPCONFIRM function your program can continue processing.

Request-to-Write Function

Your program uses the **request-to-write** (RQSWRT) function to indicate that it wants to send something to the remote program (or it wants to end the session in a controlled manner) rather than continue receiving data. The remote program, however, must decide whether to stop sending and when to stop. The request-to-write function causes an APPC SIGNAL to be sent to the remote system.

After issuing a request-to-write function, your program must continue to receive data until it receives a return code that indicates the remote program is ready to begin receiving (if it decides to do so). Your program, in response to the return code, can then begin sending its data, perform other processing, or end.

Allow-Write Function

Your application program uses the **allow-write** (ALWWRT) function to inform the remote program that your program is done sending data and is ready to receive. This causes a change-direction indicator to be sent to the remote program.

After issuing the allow-write function, your program can then issue an input operation to receive data from the remote program.

Timer Function

Your program can use the **timer** function to set a timer. A read-from-invited-program-devices operation is the only operation with which your program can determine that a timer has ended. For example, your program can set a timer and invite multiple program devices. It can then issue a read-from-invited-program-devices operation, which would ensure that your program does not wait indefinitely when no data is received on any of the invited devices. The read-from-invited-program-devices operation can complete with a return code indicating that a timer has ended.

Get-Attributes Operation

Your program uses the **get-attributes** operation to determine the status of the session. It can be issued at any time during a session. The operation gets the current status information about the session to which your program is communicating. Refer to Appendix C and to the *ICF Programmer's Guide* for more information about the get-attributes operation.

Ending a Transaction Using the Detach Function

Your program uses the **detach** function to inform the remote program that your program is done sending data and has ended the transaction. The detach function indicates that the current record is the last record to be sent and causes an APPC conditional end bracket (CEB) to be sent. The detach function can be issued by either the source or target program and can only be issued when the program is in the send state.

When a detach function and a confirm function are used together, the transaction may not end when the detach request is sent if the remote program sends or has sent a negative response. The sender of the negative response is responsible for the initial error recovery. The point at which action is taken to recover from the error determines when the transaction is ended.

After a detach function is accepted by your program, no further input or output operations with the remote program are allowed.

After a target program issues a detach function, both the session and the transaction end. No further operations are valid on the program device.

Ending a Session

The following operations and function can be used by your program to end a session with the remote program.

Release Operation

Your program uses the **release** operation to attempt to end the program's attachment to a session. Depending on how the session was started, the release operation produces different results:

- If the release operation is issued by the program that is the source of the conversation, the session ends immediately (unless some error condition occurs). The operation frees the resources (allocated to the program) that were used during the session. If the release operation is not successful, the end-of-session function can be issued to end the session. The release operation is valid *only* when a transaction is not active for source programs. This means that a detach function must first be issued before the release, or that a successful evoke function was not issued on the session.
- If the release operation is issued from the program that is the target of the conversation, the connection to the program is temporarily ended, and the transaction is kept active. This target conversation can be acquired again later in the same job; however, the session is not available for use by other programs until the target program issues an end-of-session function, a detach function, or ends. The detach function cannot be issued before the release because the detach function ends the session.

End-of-Session Function

Your program uses the **end-of-session** function to end the program's attachment to a session. If a program is in a receive state, the end-of-session function causes a negative-response indication and an APPC FMH7 to be sent to the

remote system. If the program is in a send state, the end-of-session function causes an APPC FMH7 to be sent to the remote system. Unlike the release operation, the end-of-session function always ends the session (if it still exists). However, if the function is issued during an active transaction, APPC ends the transaction abnormally, and if the conversation is synchronous (see “Conversations” on page 3-1), the partner program will also be notified that the transaction ended abnormally. For example, your program could issue the end-of-session function after an error has occurred on one of its previous operations; it may be an error from which your program cannot easily recover.

When your program issues an end-of-session function, APPC ends the program’s attachment to the session and frees the resources in the AS/400 system that were used during the session. The resources are made available to other programs in the AS/400 system that want to establish a session.

Notes:

1. When your program is a source program, it should use the end-of-session function or a release operation after a transaction has ended normally.
2. When a target program, started by a program start request, receives a detach function, it should issue an end-of-session function to make the conversation available for other transactions. When a target program sends a detach function, ICF implicitly issues an end-of-session function for your program.

Close Operation

Your program uses the **close** operation to close the ICF file and to end the program’s attachment to active sessions the program has acquired. If a program is in a receive state, the close operation causes a negative-response indication and an APPC FMH7 to be sent to the remote system. If the program is in a send state, the close operation causes an APPC FMH7 to be sent to the remote system. If the **close** operation is issued from a source program, any active sessions connected to the ICF file are ended, and all resources that were allocated for the file are deallocated. If a transaction is active when the

close operation is issued, both the session and the transaction are abnormally ended.

If the close operation is issued from the program that is the target of the conversation, the connection to the program is only temporarily ended, and the transaction is kept active. This target conversation can be acquired again later in the same job by issuing the **open** operation and the **acquire** operation. However, the session is not available for use by other programs until the target program issues an end-of-session function, a detach function, or ends.

Using Response Indicators

Response indicators are defined to your program in the ICF file and are set on each input operation. However, these indicators are optional and major and minor return codes can also be used to indicate the status of input operations.

Receive-Confirm

Your program uses the **receive-confirm** (RCVCONFIRM) response indicator to receive an indication that the record received from the remote system contains a confirm request. A received confirm request indicates that the other program is expecting your program to perform a specific action to synchronize the programs. This action can be a RSPCONFIRM function to positively respond; or a close operation, FAIL function, or EOS function to negatively respond.

The presence of the confirm request is also indicated by major return codes 00 (data received), 02 (end job or end subsystem in progress), or 03 (no data received) and minor return codes 13, 14, 15, 18, 1C, 1D, 44, 45, and 46.

Receive-Fail

Your program uses the **receive-fail** (RCVFAIL) response indicator to receive an indication that the partner application has detected an error. Your application program should take the appropriate recovery action. Your program remains in receive state after receiving the RCVFAIL indicator and should continue to issue input operations. Receipt of a fail request is also indicated

by a major and minor return code. Refer to return codes 83C7 through 83CC in Appendix B.

The failure notification is always received without user data. Receipt of the failure causes a notify message to be sent if the RCVFAIL keyword is not specified. The same messages are used whether the program is in send or receive state.

Receive-Turnaround

Your program uses the **receive-turnaround** (RCVTRNRND) response indicator to receive an indication that the record returned in the input buffer of the local system has ended a transmission. The RCVTRNRND response indicator indicates that the remote program is ready to receive data.

Your program can detect a receive-turnaround indicator by checking for a major return code of 00 (user data received), 02 (end job or end subsystem in progress), or 03 (no data received) and minor return codes of 00, 04, 06, 13, 14, and 44.

Receive-Detach

Your program uses the **receive-detach** (RCVDETACH) response indicator to receive an indication that a detach request has been received. If the detach request is received along with data, major return code 00 is set along with the appropriate minor code. If the detach request is received without data, major return code 03 is set along with the appropriate minor code. The detach request can also be received with a major return code of 02 (end job or end subsystem in progress), along with the appropriate minor code. The minor code can be 08, 0C, 11, 1C, or 46.

Receive-Control-Data

Your program uses the **receive-control-data** (RCVCTLDTA) response indicator to receive an indication that the record retained in the input buffer contains control data. If the receive-control-data response indicator is received along with data, major return code 00 is set along with the appropriate minor code. If the response indicator is received without data, major return code 03 is set along with the appropriate minor

code. The response indicator can also be received with a major return code of 02 (end job or end subsystem in progress), along with the appropriate minor code. The minor return code can be 02, 04, 05, 06, 0C, 11, 13, 18, 1D, 44, 45, 46, 3421, or 3481.

Using Input/Output Feedback Areas

Your program may have access to the input/output (I/O) feedback area. If it does, there are certain fields that you should be aware of when writing APPC applications.

Major return code

This field contains the major return code indicating the status of input and output operations.

Minor return code

This field contains the minor return code indicating the status of input and output operations.

Negative-response data

For some return codes, this field contains the SNA sense data received from the remote system. Refer to the *APPN Guide* and to the *SNA Formats* manual for more information about sense data.

Request-to-write indicator

This field indicates whether the remote program has requested permission to send data.

Format name data

This field contains the APPC map name received from the remote system. If the remote system is an AS/400 system, this is the record format name sent as a result of a write operation with the FMTNAME function specified.

Mode

This field contains the name of the mode to which the session is attached. Refer to "Mode Description" on page 2-3 for a description of modes.

Actual received data length

This field contains the length of the data received on an input operation.

Length of the record associated with the last I/O operation

This field contains the length of the record associated with the last I/O operation. This value includes data, option indicators, response indicators, and program-to-system data, if applicable.

Using Return Codes

After each operation, an ICF return code is returned to your program. Your program should check this return code to determine:

- The status of the operation just done
- The operation that should be done next

For example, a major return code of 00 indicates that data was received on an input operation. Along with this major code you can receive from APPC minor codes, such as:

- 01: Indicates that your program should continue receiving data.
- 08: Indicates that the remote program has ended the transaction. Your program can do one of the following:
 - If it is a source program, issue another evoke function or end the session.
 - If it is a target program, end the session and continue local processing or go to end of job.
- 1C: Indicates that the remote program has ended the transaction and requested confirmation. Your program must first respond either positively or negatively to the confirmation request. If your program responds positively, it should continue as for the 08 minor code. If it responds negatively, it should then inform the remote program why it responded negatively or it can go to end of job without performing error recovery. In any case, if your program responds negatively, it is responsible for the appropriate error recovery.

Another example would be a major code of 83. In this case either the local system, remote system, or remote program has detected an error that may be recoverable. Different minors can be returned just as for the 00 major. For example:

- If your program receives a C7 minor return code, the remote program has sent your program an error condition (on the AS/400 system this is done by using the fail function). If your program receives this return code, or any of the other fail return codes, it should perform an input operation to allow the remote program to send the appropriate error recovery information. In this situation, the remote program is responsible for the necessary error recovery.
- If your program receives a CD minor return code, your program has issued a confirm function which is currently not allowed. Either your program is using a transaction which was not started with a synchronization level of confirm or the current state of the transaction does not allow the confirm function. For this return code, your program is responsible for the necessary error recovery. The session and transaction are still active and you can recover from this error by issuing the operation without the confirm function.

Your program should check the ICF return codes at the completion of every operation to ensure that the operation completed successfully or that the appropriate recovery action is taken.

Refer to Appendix B for a description of the return codes that can be returned to your application when it is using APPC.

Mapping between ICF Operations and Functions and LU Type 6.2 Verbs

The two tables in this section map ICF operations and functions to LU type 6.2 verbs, and can be useful if you are writing applications between an AS/400 system and another system that supports APPC. A more detailed mapping of the architected verbs to the APPC implementation is provided in Appendix C.

Mapping of LU Type 6.2 Verbs to ICF Operations and Functions

The following table lists the LU type 6.2 verbs and the corresponding ICF operation or function, the DDS keyword, and the system-supplied format.

Figure 5-4 (Page 1 of 2). Mapping between ICF Operations or Functions and LU Type 6.2 Verbs

Mapped or Basic Conversation Verbs	ICF Operation or Function	DDS Keyword	System-Supplied Format
MC_ALLOCATE or ALLOCATE	Combination of acquire operation and evoke function	EVOKE	\$\$\$EVOKNI
MC_CONFIRM or CONFIRM	Confirm function	CONFIRM	No equivalent
MC_CONFIRMED or CONFIRMED	Respond-to-Confirm function	RSPCONFIRM	No equivalent
MC_DEALLOCATE or DEALLOCATE	Write operation with detach function, and, depending on the type of program, one of the following: <ul style="list-style-type: none"> • For a source program, an end-of-session function, release, or close operation must follow the detach. • For a target program, a close operation should follow the detach. 	DETACH EOS	\$\$\$SENDET \$\$\$EOS
MC_FLUSH or FLUSH	Force-data function	FRCDTA	No equivalent
MC_GET_ATTRIBUTES or GET_ATTRIBUTES	Get-attributes operation	Not applicable	Not applicable
MC_POST_ON_RECEIPT or POST_ON_RECEIPT	Invite function	INVITE	\$\$\$SEND with 0 length
MC_PREPARE_TO_RECEIVE or PREPARE_TO_RECEIVE	Allow-write or invite function	ALWWRT or INVITE	\$\$\$SEND with 0 length
MC_RECEIVE_AND_WAIT or RECEIVE_AND_WAIT	Read operation to a specific program device	Not applicable	Not applicable
MC_REQUEST_TO_SEND or REQUEST_TO_SEND	Request-to-write function	RQSWRT	\$\$RCD Note: This also causes an invite to be issued.
SEND_DATA	Write operation of a format with a data length greater than 2	Not applicable	\$\$\$SENDNI
MC_SEND_DATA	Write operation	Not applicable	\$\$\$SENDNI
MC_SEND_DATA with USER_CONTROL_DATA	Write operation with CTLDTA keyword	CTLDTA	Not applicable
MC_SEND_ERROR or SEND_ERROR	Fail function	FAIL	\$\$FAIL

Figure 5-4 (Page 2 of 2). Mapping between ICF Operations or Functions and LU Type 6.2 Verbs

Mapped or Basic Conversation Verbs	ICF Operation or Function	DDS Keyword	System-Supplied Format
MC_TEST or TEST (REQUEST_TO_SEND_RECEIVED)	No equivalent; however, the application program can check the request-to-write indicator in the I/O feedback area to determine if a REQUEST_TO_SEND was received.	Not applicable	Not applicable
MC_TEST or TEST (POSTED)	Get-attributes operation	Not applicable	Not applicable
GET_TYPE	Get-attributes operation	Not applicable	Not applicable
WAIT	Read-from-invited-program- devices operation	Not applicable	Not applicable

Example ICF Operations and Functions Mapped to LU Type 6.2 Verbs

The following table provides some example ICF operations and functions that can be used in your application program and the LU type 6.2 architected verb and parameter to which it corresponds. This table only provides DDS keyword examples, and should be used to supplement the more detailed information contained in Appendix C.

Notes:

1. Brackets [] indicate that contents are optional.
2. Parameters that are required (such as RESOURCE and RETURN_CODE) are not always listed explicitly in the table.
3. These examples show one way of performing the operation or function; there may be others.

Figure 5-5 (Page 1 of 3). Mappings for Example ICF Operations and Functions

ICF Operations/Functions	LU Type 6.2 Verbs-Basic or Mapped
<p>An acquire operation followed by: EVOKE([library-name/]program-name)</p> <p>Note: It is the combination of an acquire operation and evoke function that corresponds to [MC_]ALLOCATE. With an evoke function, you can also specify any of the following:</p> <p>[parameter-1...[parameter-255]]</p> <p>SECURITY(3 *USER) or SECURITY([1 profile ID] [2 password] [3 userid])</p> <p>SYNLVL(*NONE)</p> <p>DFREVOKE</p> <p>SYNLVL(*CONFIRM) ⋮ Write (data length > 0) with CONFIRM</p>	<p>Corresponds to:</p> <p>[MC_]ALLOCATE with TPN(program-name) followed by a [MC_]FLUSH</p> <p>[PIP(YES(variable1 variable2...variablen))]</p> <p>SECURITY(SAME) or SECURITY(PGM)</p> <p>SYNC_LEVEL(NONE)</p> <p>Do not follow the [MC_]ALLOCATE with a [MC_] FLUSH</p> <p>SYNC_LEVEL(CONFIRM) ⋮ [MC_]SEND_DATA followed by [MC_]CONFIRM</p>
<p>If your program specifies SYNLVL(*CONFIRM) and the CONFIRM DDS keyword, the remote program must then issue a:</p> <p>RSPCONFIRM (positive response) or FAIL (negative response)</p> <p>If the remote program issues a Get-attributes operation, it can determine the SYNLVL established by the source program.</p>	<p>Corresponds to:</p> <p>[MC_]CONFIRMED or [MC_]SEND_ERROR</p> <p>[MC_]GET_ATTRIBUTES</p>
<p>Your program issues a: FRCDTA</p>	<p>Corresponds to: [MC_]FLUSH</p>
<p>Your program issues a write with FMTNAME to send the record format name along with data to the remote system.</p> <p>Note: The format-name can only be used with mapped conversations, and should only be used if the remote system is able to receive the APPC architected map name GDS ID variable.</p>	<p>Corresponds to: MC_SEND_DATA with [MAP_NAME(YES(map-name))]</p>
<p>Your program uses a: Write operation to send data to the remote system, and then uses a: Read operation to receive data from a specific program device (for example, PGMDEVA) and waits to receive data before continuing</p>	<p>Corresponds to: [MC_]SEND_DATA</p> <p>[MC_]RECEIVE_AND_WAIT</p>
<p>Your program uses a: Write operation with a CTLDTA keyword</p>	<p>Corresponds to: [MC_]SEND_DATA with USER_CONTROL_DATA</p>

Figure 5-5 (Page 2 of 3). Mappings for Example ICF Operations and Functions

ICF Operations/Functions	LU Type 6.2 Verbs-Basic or Mapped
<p>If your program issues a: Write without data and INVITE</p> <p>requesting data from the remote program to send data for a program device (for example, PGMDEVA), your program receives control without waiting for input. Later, your program can issue a Read-from-invited-program-devices operation.</p> <p>A successful completion return code (set in the I/O feedback area) tells your program that data has been received and is in your program's buffer.</p>	<p>Corresponds to: [MC_]POST_ON_RECEIPT or [MC_]PREPARE_TO_RECEIVE followed by [MC_]POST_ON_RECEIPT, if the program is in send state</p> <p>WAIT</p>
<p>Your program issues a: Write without data and RQSWRT</p> <p>to indicate that it wants to send data to the remote program rather than continue receiving data.</p>	<p>Corresponds to: [MC_]REQUEST_TO_SEND</p>
<p>Your program issues a: Write with data and ALWWRT</p> <p>to inform the remote program that your program is done sending data and is ready to receive.</p>	<p>Corresponds to: [MC_]SEND_DATA followed by [MC_]PREPARE_TO_RECEIVE</p>
<p>Your program issues a: Write without data and ALWWRT</p> <p>to inform the remote program that your program is done sending data and is ready to receive.</p>	<p>Corresponds to: [MC_]PREPARE_TO_RECEIVE</p>
<p>To end a session, a source program issues: Write with DETACH, followed by EOS or close</p> <p>Note: The combination of DETACH and EOS corresponds to [MC_]DEALLOCATE. The detach function indicates that the record is the last record to be sent and causes an APPC CEB to be sent.</p>	<p>Corresponds to: [MC_]DEALLOCATE with TYPE(FLUSH)</p>
<p>If your program issues: DETACH with CONFIRM, followed by EOS or close,</p> <p>the transaction does not end if the remote program sends a FAIL (negative acknowledgement)</p> <p>Otherwise, the transaction ends normally when the remote program issues a: Write without data and RSPCONFIRM</p>	<p>[MC_]DEALLOCATE with TYPE(CONFIRM)</p> <p>[MC_]SEND_ERROR</p> <p>[MC_]CONFIRMED</p>
<p>If the program is a target program, started by a program start request, and it receives a detach indication using: RCVDETACH</p> <p>(or it receives return codes 0008, 0208, or 0308), it should end the session using: EOS</p>	<p>Corresponds to: RETURN_CODE(DEALLOCATE_NORMAL) on an appropriate LU 6.2 operation</p> <p>[MC_]DEALLOCATE with TYPE(LOCAL)</p>

Figure 5-5 (Page 3 of 3). Mappings for Example ICF Operations and Functions

ICF Operations/Functions	LU Type 6.2 Verbs-Basic or Mapped
<p>If the program is a source program, and it receives a detach indication using: RCVDETACH</p> <p>(or it receives return codes 0008, 0208, or 0308), it should end the session using: a release operation followed by a close operation</p> <p>(An end-of-session function is not necessary for the source program.)</p>	<p>Corresponds to:</p> <p>RETURN_CODE(DEALLOCATE_NORMAL) on an appropriate LU 6.2 operation</p> <p>[MC_]DEALLOCATE with TYPE(LOCAL)</p>
<p>If your program uses an EOS</p> <p>during an active transaction, both the transaction and the conversation are ended abnormally by APPC.</p>	<p>Corresponds to:</p> <p>[MC_]DEALLOCATE with TYPE(ABEND)</p>

Flow Diagrams

This section provides example flow diagrams showing how two programs can exchange information and data. The first flow diagram, showing how ICF operations and functions are used, is based on the inquiry application examples in Appendix E. In this example, a source program on a local AS/400 system requests information about a part number from a target program on a remote AS/400 system.

The second flow diagram, showing how the LU type 6.2 verbs and parameters are used, parallels the flow diagram for inquiry applications.

A Note about the Flow Diagrams

In the flow diagrams in this section, vertical dotted lines indicate the components involved in the exchange of information between systems. The horizontal arrows indicate the direction of the flow for that step. The numbers lined up on the left side of the flow are reference points to the flow and indicate the progression of verbs or functions made on the conversation. These same numbers correspond to the numbers under the **Step** heading of the text description for each example.

Not all possible parameters that can be issued with a verb are listed in the flow diagrams. For a complete list and description of the parameters, refer to Appendix C.

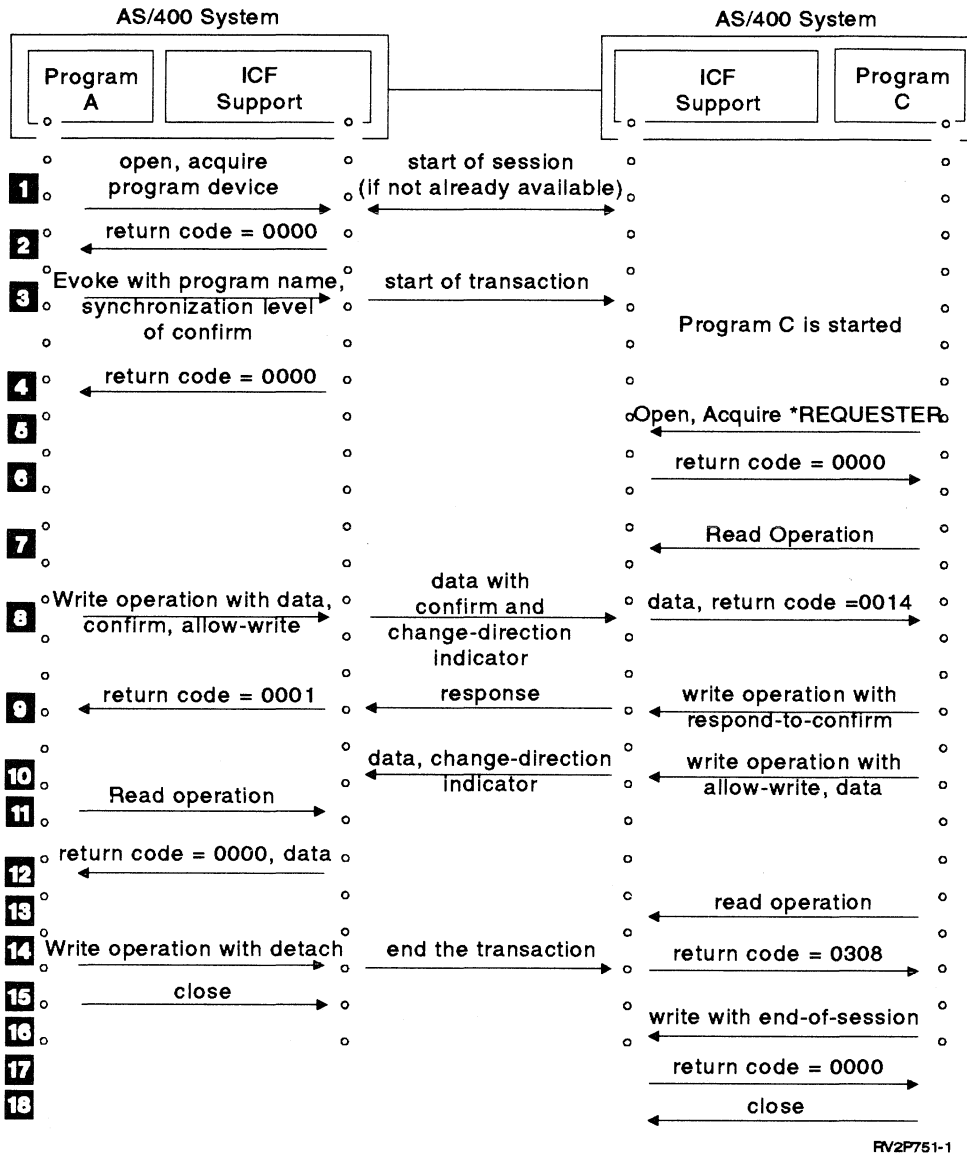
Flow Diagram for Inquiry Applications Using ICF Operations and Functions

The steps shown in Figure 5-7 are described below.

Figure 5-7 on page 5-20 shows the flow for the exchange of information that occurs in an inquiry application using ICF operations and functions.

Figure 5-6. Description of Flow Diagram

Step	Description
1 and 2	Program A opens an ICF file and issues an acquire operation to start a session with the remote AS/400 system. This causes a session to be established if a session is not already available. The major and minor return code of 0000 indicates that the acquire operation was successful.
3 and 4	Program A issues an evoke function, which establishes a conversation with the partner program. The evoke function is issued with a synchronization level of confirm.
5 and 6	Program C opens an ICF file on the remote system and issues an acquire operation for the requesting program device to establish a logical connection to the session and transaction.
7	Program C issues a read operation to receive information from Program A.
8	Program A issues a write operation with a confirm function and an allow-write function to request information about a part number (sent as the data). When these functions are used, the data is flushed, the change-direction indicator is sent, and a confirmation request is sent to the partner program. The read operation that program C issued (step 7) completes with a major and minor return code of 0014, indicating that data was received with the change-direction indicator, and the partner program requested confirmation.
9	Program C must now respond to the confirmation request. Program C issues a write operation with the respond-to-confirm function. As a result, a positive response is sent to the received confirmation request. Program C prepares to send data. Meanwhile, Program A receives a major and minor return code of 0001 in response to the confirmation request, indicating that the write operation it issued completed successfully.
10	Program C sends the requested data using the write operation with the allow-write function.
11 and 12	Program A issues a read operation and receives the data. The major and minor return code of 0000 on the read operation indicates that data was received with the change-direction indicator. Program A can send more requests to be processed, if necessary.
13	Program C issues a read operation to receive information from Program A.
14 and 15	Program A ends the transaction by issuing a detach function and closing the ICF file. The read operation that Program C issued (step 13) completes with a major and minor return code of 0308, indicating that a detach was received without any data.
16 and 17	Though Program C can continue local processing, it ends the session using the end-of-session function.
18	Program C then closes the ICF file.



RV2P751-1

Figure 5-7. Data Flow Using ICF Operations and Functions

Flow Diagram for Inquiry Applications Using LU Type 6.2 Verbs

Figure 5-9 on page 5-22 shows the flow for the exchange of information that occurs between two systems using LU type 6.2 verbs, and parallels the flow diagram for the ICF inquiry applications on 5-19.

Mapped conversation verbs are used as examples in the flow diagram, but basic conversation verbs could also be used.

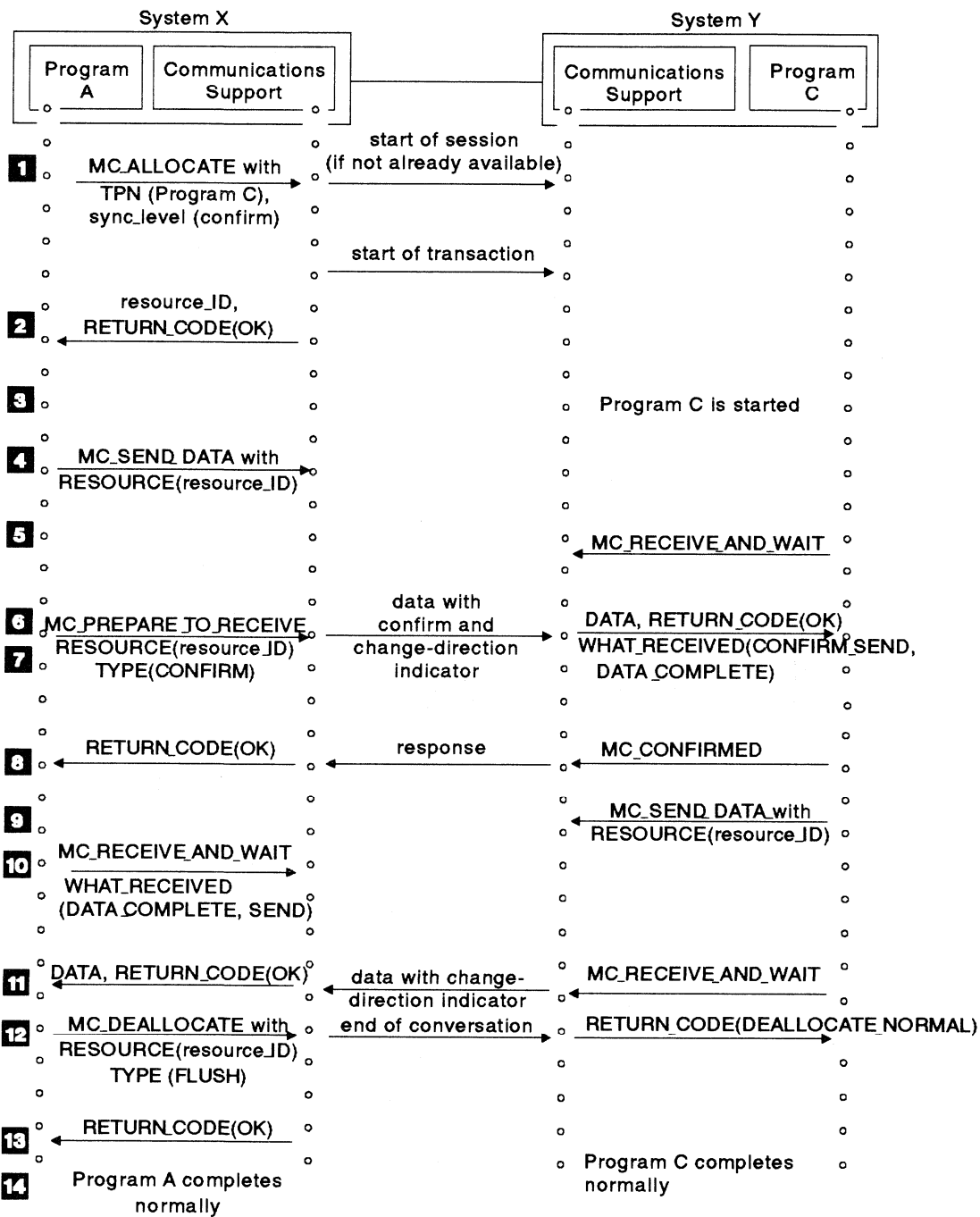
The steps shown in Figure 5-9 are described as follows:

Figure 5-8 (Page 1 of 2). Description of Flow Diagram

Step	Description
1	The MC_ALLOCATE verb allocates a session between System X and System Y, and on that session allocates a conversation between the local transaction program and a remote transaction program. The TPN parameter specifies the name of the remote program to be connected at the other end of the conversation. If a session between the logical units (LUs) is not already available, a session is activated.

Figure 5-8 (Page 2 of 2). Description of Flow Diagram

Step	Description
2	No errors occurred on the MC_ALLOCATE, so the RETURN_CODE is set to OK. A <i>resource_ID</i> is also assigned to the conversation.
3	Program C is started, and Program A and Program C can now have a conversation.
4	Program A uses the MC_SEND_DATA verb to send data to the remote program, using the <i>resource_ID</i> assigned to the conversation. Data is buffered and is not sent to Program C until Program A issues the MC_PREPARE_TO_RECEIVE with TYPE(CONFIRM) (see step 6).
5	Program C uses the MC_RECEIVE_AND_WAIT verb to wait for information to arrive on the specified conversation, but it does not receive information until Program A issues its next verb (see step 6). The information can be data, conversation status, or a request for confirmation.
6 and 7	<p>Program A uses the MC_PREPARE_TO_RECEIVE verb with TYPE(CONFIRM) to indicate that a confirmation is requested, and that the mapped conversation should change from a send to a receive state. The RESOURCE parameter specifies the <i>resource_ID</i> of the conversation on which data is to be sent. As a result of carrying out the CONFIRM function on this verb, the LU flushes its send buffer, and data along with the confirm and change-direction indicator flows across the line to the other system. Data is received by Program C, and a RETURN_CODE(OK) indicates that no error was encountered.</p> <p>The WHAT_RECEIVED parameter is returned to Program C with DATA_COMPLETE and CONFIRM_SEND specified, which indicates that Program A has issued an MC_PREPARE_TO_RECEIVE with TYPE(CONFIRM) and is requesting Program C to respond.</p> <p>Note: The AS/400 system, for performance reasons, returns multiple values for WHAT_RECEIVED.</p>
8	Program C responds by issuing an MC_CONFIRMED verb. The RETURN_CODE(OK) indication, which is returned to Program A, indicates that Program C received data without errors.
9	Program C sends data to Program A using the MC_SEND_DATA verb (in response to an inquiry by Program A). The data is buffered and does not flow across the line until Program C issues an MC_RECEIVE_AND_WAIT (see step 11).
10	Program A uses the MC_RECEIVE_AND_WAIT verb to wait for information to arrive on the specified conversation.
11	Program C issues the MC_RECEIVE_AND_WAIT verb to wait for information to arrive on the specified conversation. As a result, the buffered data along with the change-direction indicator flows across the line. RETURN_CODE(OK) returned to Program A indicates no errors were encountered when the data was received.
12	Program A issues an MC_DEALLOCATE verb with TYPE(FLUSH) to release the conversation from the remote program. TYPE(FLUSH) flushes the local data buffer of the LU and deallocates the conversation normally. The <i>resource_ID</i> that was assigned to this conversation is no longer assigned when deallocation is complete. The RETURN_CODE(DEALLOCATE_NORMAL) tells Program C that the conversation is deallocated.
13 and 14	The RETURN_CODE(OK) tells Program A that the conversation is successfully deallocated. Both Program A and Program C complete normally.



RV2P752-1

Figure 5-9. Data Flow Using LU Type 6.2 Verbs

Chapter 6. Writing APPC Application Programs Using CPI Communications

CPI Communications provides an application programming interface for applications that use program-to-program communications. The interface uses the SNA LU 6.2 architecture to perform the following services:

- Establishing a conversation (and its characteristics)
- Sending and receiving data
- Exchanging control information
- Ending a conversation
- Notifying a partner of errors in the communication.

This chapter contains the following information:

- A description of communications side information, which stores the required initialization parameters that allow the CPI Communications program to establish a conversation with its partner program.
- How to create and manage communications side information.
- A listing of the AS/400 CPI Communications calls, which allow a CPI Communications program to communicate with its partner program.

Note: AS/400 does not support synchronization point services.

- A mapping of CPI Communications calls to ICF operations and functions.
- An example of a data flow using AS/400 CPI Communications.
- A listing of CPI Communications return codes.

For more detailed information about writing CPI Communications applications programs, see the *CPI Communications Reference*.

Description of Communications Side Information

For a program to establish a conversation with a partner program, CPI Communications requires initialization parameters, such as the name of the partner program and the name of the LU at

the node of the partner program. These may be stored in the **communications side information**.

Your program must specify the communications side information name as the symbolic destination name parameter on the Initialize_Conversation call to use the stored characteristics. If your program does not specify a communications side information name (that is, the *sym_dest_name* is eight space characters), your program must issue the Set_Partner_LU_Name and Set_TP_Name calls before issuing the Allocate call. Also, when no communications side information name is specified, a *mode_name* of eight space characters is used.

On the AS/400 system, the communications side information is a system object of type *CSI (communications side information). It contains information that defines the route to the remote system, such as the RMTLOCNAME, RMTNETID, and MODE parameters.

The AS/400 communications side information object also contains additional information: the device description, the local location name (local LU name), and the authority. The remote network ID, remote location name, device description, and local location name are used by the AS/400 system to determine the route to the remote system. The additional parameters allow you to use the full support of APPN while using CPI Communications. Refer to "Using the Location Parameters" on page 3-4 for a detailed description of how the AS/400 system uses these parameters.

Figure 6-1 on page 6-2 describes the information contained in the communications side information object and maps it to the CPI Communications counterparts.

Figure 6-1. Description of Communications Side Information Object

AS/400 System Parameters	Description
Remote location name and Remote network ID	The name of the logical unit (LU) on the remote system. These parameters correspond to the <i>partner_LU_name</i> , which is defined by the CPI Communications architecture as a required characteristic for the communications side information. A fully-qualified <i>partner_LU_name</i> is defined as the network ID concatenated by a period with the network LU name (that is, network ID.network LU name). On the AS/400 system, the remote network ID is the network ID, and the remote location name is the network LU name.
Mode	The name of the mode used to control the session. It is used to designate the properties for the session that will be allocated for the conversation, such as the class-of-service parameter to be used on the conversation. This parameter corresponds to the CPI Communications <i>mode_name</i> characteristic. To use a <i>mode_name</i> of eight blank characters, you must use the special value of BLANK. The AS/400 system does not support sending a <i>mode_name</i> of 'BLANKbbb' to a remote system.
Device description	The name of the device description, which describes the characteristics of the logical connection between a local and remote location. This AS/400 parameter further qualifies the route defined by the remote location name and the remote network ID.
Local location name	The name of the local location, which specifies the name of your local network LU name in a network. This AS/400 parameter further qualifies the route defined by the remote location name and the remote network ID.
Program name	Name of the target program that is to be started. This corresponds to the <i>TP_name</i> , which is defined by the CPI Communications architecture as a required characteristic for the communications side information.
Authority	The authority given to users who do not have specific authority to the communications side information object. This is an AS/400 system parameter.

Managing the Communications Side Information

To manage the communications side information object, the AS/400 system provides CL commands (as well as command prompting from a display) that allow you to create, display, print, change, delete, and work with the communications side information. The following is a list of the CL commands you can use to manage the communications side information object:

CRTCSI

Used to create the CPI Communications side information object.

CHGCSI

Used to change the CPI Communications side information object.

DSPCSI

Used to display or print the CPI Communications side information object.

DLTCSI

Used to delete the CPI Communications side information object.

WRKCSI

Provides a menu from which the user can create, change, display, delete, or print the CPI Communications side information object.

Specifying the Communications Side Information Commands: The following describes the parameters for the CRTCSI and CHGCSI commands and lists the values for each parameter. Refer to "Using the Location Parameters" on page 3-4 for more information on how the AS/400 system uses the RMTNETID, RMTLOCNAME, DEV, LCLLOCNAME, and MODE parameters.

CSI

Specifies the name of the communications side information object. An object name must be specified.

side-information-name: Specify the name of the object that will contain the desired communications side information object. This is the name an application uses for the *sym_dest_name* on an Initialize_Conversation (CMINIT) call.

The possible library values are:

***CURLIB:**

The current library for the job is used to create the communications side information object. If no library is specified as the current library for the job, the QGPL library is used.

library-name: Specify the name of the library in which the communications side information object will be stored.

RMTLOCNAME

Specifies the remote location name associated with the symbolic destination name.

This name is the logical unit on the remote system and corresponds to the second part of the CPI Communications

partner_LU_name. This is a required parameter.

remote-location-name: Enter the name of the remote location that should be associated with the symbolic destination name.

TNSPGM

Specifies the name (up to 64 characters) of the transaction program to be started. This is a required parameter.

transaction-program-name: Specify the transaction program name.

Notes:

1. If a program start request is received on the AS/400 system with an unqualified program name, (that is, it has no library name) the system uses the library list specified on the QUSRLIBL system value at the time the subsystem that is handling the program start request was started.
2. If a program start request is received on the AS/400 system with a qualified program name, the program name can be in the form 'program.library' or in the form 'library/program'.
3. Program names and library names on the AS/400 system are limited to 10 characters each.
4. To specify the SNA service transaction program names, enter the hexadecimal representation of the service transaction program name. For example, to specify a service transaction program name in

which the hexadecimal representation is 21F0F0F1, you would enter X'21F0F0F1'.

DEV

Specifies the name of the device description used for the remote system.

The possible values are:

***LOC:** The device is determined by the system.

device-name: Specify the name of the device that is associated with the remote location.

LCLLOCNAME

Specifies the name of your location.

The possible values are:

***LOC:** The local location name is determined by the system.

***NETATR:** The local location name that is in the network attributes is used.

local-location-name: Specify the name of your location. Specify the local location if you want to indicate a specific local location name for the remote location.

MODE

Specifies the mode used to control the session. This name is the same as the CPI Communications *mode_name.*

The possible values are:

***NETATR:** The mode specified in the system network attributes is used.

BLANK: The *mode_name*, consisting of 8 blank characters, is used.

mode-name: Specify the *mode_name* for the remote location.

Note: The values SNASVCMG and CPSVCMG are not allowed.

RMTNETID

Specifies the remote network ID used with the remote location. The CPI Communications *partner_LU_name*, which consists of the remote network identifier and the remote location, determines the logical unit in the network.

The possible values are:

***LOC:** The remote network ID for the remote location is used.

***NETATR:** The remote network ID specified in the network attributes is used.

*NONE: The remote network has no name.

remote-network-ID: Specify a remote network ID.

AUT

Specifies the authority you have given to users who do not have specific authority to the object, who are not on the authorization list, and whose users' group has no specific authority to the object.

The possible values are:

*LIBCRTAUT:

Public authority for the object is taken from the CRTAUT parameter of the specified library. This value is determined at create time. If the CRTAUT value for the library changes after the object is created, the new value does not affect any existing objects.

*CHANGE:

Change authority allows the user to perform all operations on the object except those limited to the owner or controlled by object existence authority and object management authority. The user can change the object and perform basic functions on the object. Change authority provides object operational authority and all data authority.

*ALL:

All authority allows the user to perform all operations on the object except those limited to the owner or controlled by authorization list management authority. The user can control the object's existence, specify the security for the object, change the object, and perform basic functions on the object. The user cannot transfer ownership of the object.

*USE:

Use authority allows the user to perform basic operations on the object, such as run a program or read a file. The user is prevented from changing the object. Use authority provides object operational authority and read authority.

*EXCLUDE:

Exclude authority prevents the user from accessing the object.

authorization-list: Specify the name of the authorization list whose authority is

used for the communications side information.

TEXT

Specify text that briefly describes this object and its function

The possible values are:

***BLANK**: No text is specified.

'*description*': Specify no more than 50 characters, enclosed in apostrophes.

Using Program Calls

CPI Communications programs communicate with each other by making **program calls**, which are used to establish the characteristics of the conversation and to exchange data and control information between the programs. Default characteristics for a conversation are established when a source program issues a call to start a conversation with a target program. For example, the default value for the *conversation_type* characteristic is CM_MAPPED_CONVERSATION. The *CPI Communications Reference* provides a table of all the characteristics and their default values.

When a program makes a CPI Communications call, the program passes characteristics, data, and control information to CPI Communications using input parameters. When the call completes, CPI Communications passes characteristics, data, and status information back to the program using output parameters. The *return_code* parameter, for example, is returned for all program calls. It indicates whether a call completed successfully or if some error was detected that caused the call to fail.

CPI Communications uses additional output parameters on some calls to pass status and control information to the program. These parameters include the *request_to_send_received* indicator, the *data_received* indicator, and the *status_received* indicator.

Program calls can be made from application programs written in the following high-level programming languages:

- C/400
- COBOL/400

- Cross System Product (CSP) (implements the Application Generator common programming interface)
- FORTRAN/400
- REXX
- RPG/400

The general call format is to show the name of the CPI Communications call and the parameters used. An example of the format is:

```
CALL CMPROG(parm0, parm1, parm2, . . . parmN)
```

where CMPROG is the name of the call and parm0, parm1, parm2, and parmN represent the parameter list described in the individual call description.

This format would be translated into the following syntax for each of the following languages:

C/400

```
CMPROG (parm0,parm1,parm2,...parmN)
```

COBOL/400

```
CALL "CMPROG" USING  
parm0,parm1,parm2,...parmN
```

CSP

```
CALL CMPROG parm0,parm1,parm2,...parmN
```

FORTRAN/400

```
CALL CMPROG (parm0,parm1,parm2,...parmN)
```

REXX

```
ADDRESS CPICOMM 'CMPROG parm0 parm1 parm2  
... parmN'
```

RPG/400

```
CALL 'CMPROG' plist
```

Program Calls Supported by the AS/400 System

The CPI Communications program calls that the AS/400 system supports, along with their corresponding pseudonyms and a brief description of each call, are listed in Figure 6-2. **Pseudonyms** for program calls are used to enhance understanding and readability. For example, Send_Data is the pseudonym for the program call CMSEND, which is used by a program to send information to a conversation partner. Note that any phrase that contains an underscore in its name is a pseudonym.

Figure 6-2 (Page 1 of 2). Supported CPI Communications Program Calls

Call Name	Pseudonym	Description
CMACCP	Accept_Conversation	Used by a program to accept an incoming conversation
CMALLC	Allocate	Used by a program to establish a conversation
CMCFM	Confirm	Used by a program to send a confirmation request to its partner
CMCFMD	Confirmed	Used by a program to send a confirmation reply to its partner
CMDEAL	Deallocate	Used by a program to end a conversation
CMECT	Extract_Conversation_Type	Used by a program to view the current <i>conversation_type</i> conversation characteristic
CMEMN	Extract_Mode_Name	Used by a program to view the current <i>mode_name</i> conversation characteristic
CMEPLN	Extract_Partner_LU_Name	Used by a program to view the current <i>partner_LU_name</i> conversation characteristic
CMESL	Extract_Sync_Level	Used by a program to view the current <i>sync_level</i> conversation characteristic
CMFLUS	Flush	Used by a program to flush the send buffer of the LU
CMINIT	Initialize_Conversation	Used by a program to initialize the conversation characteristics
CMPTR	Prepare_To_Receive	Used by a program to change a conversation from send to receive state in preparation to receive data
CMRCV	Receive	Used by a program to receive data
CMRTS	Request_To_Send	Used by a program to notify its partner that it would like to send data

Figure 6-2 (Page 2 of 2). Supported CPI Communications Program Calls

Call Name	Pseudonym	Description
CMSCT	Set_Conversation_Type	Used by a program to set the <i>conversation_type</i> conversation characteristic
CMSDT	Set_Deallocate_Type	Used by a program to set the <i>deallocate_type</i> conversation characteristic
CMSD	Set_Error_Direction	Used by a program to set the <i>error_direction</i> conversation characteristic
CMSSEND	Send_Data	Used by a program to send data
CMSERR	Send_Error	Used by a program to notify its partner of an error that occurred during the conversation
CMSF	Set_Fill	Used by a program to set the <i>fill</i> conversation characteristic
CMSLD	Set_Log_Data	Used by a program to set the <i>log_data</i> conversation characteristic
CMSMN	Set_Mode_Name	Used by a program to set the <i>mode_name</i> conversation characteristic
CMSPLN	Set_Partner_LU_Name	Used by a program to set the <i>partner_LU_name</i> conversation characteristic
CMSPTR	Set_Prepare_To_Receive_Type	Used by a program to set the <i>prepare_to_receive_type</i> conversation characteristic
CMSRC	Set_Return_Control	Used by a program to set the <i>return_control</i> conversation characteristic
CMSRT	Set_Receive_Type	Used by a program to set the <i>receive_type</i> conversation characteristic
CMSL	Set_Sync_Level	Used by a program to set the <i>sync_level</i> conversation characteristic
CMSST	Set_Send_Type	Used by a program to set the <i>send_type</i> conversation characteristic
CMSTPN	Set_TP_Name	Used by a program to set the <i>TP_name</i> conversation characteristic
CMTRTS	Test_Request_To_Send_Received	Used by a program to determine whether or not the remote program is requesting to send data

Using Pseudonyms When Writing Applications

CPI Communications provides a set of system files that can be used to define pseudonyms for the characteristics, variables, and parameters needed by a CPI Communications application.

Pseudonyms enhance the readability of the program text and provide consistency in the naming of characteristics, variables, and parameters that make up the call interface. For example, instead of stating that the variable

return_code is set to an integer value of 0, it is shown to be set to a pseudonym value of CM_OK.

Pseudonyms can also be used for integer values in program code by making use of equate or define statements. On the AS/400 system, pseudonym files for the supported high-level languages are included in the library for the language you are using. Figure 6-3 on page 6-7 shows the name of the member, the source physical file, and the library containing the pseudonym files for each high-level language supported by CPI Communications on the AS/400 system.

Figure 6-3. Pseudonym Files for High-Level Languages

Language	Member Name	Source Physical File	Library
C/400	CMC	H	QCC
COBOL/400	CMCOBOL	QILBINC	QLBL
FORTRAN/400	CMFORTRN	QIFOINC	QFTN
RPG/400	CMRPG	QIRGINC	QRPB

Notes:

1. No pseudonym files are provided for CSP or REXX languages on the AS/400 system.
2. The *CPI Communications Reference* provides example pseudonym files for C/400, FORTRAN/400, RPG/400, COBOL/400, CSP, and REXX languages.

Mapping of CPI Communications Calls to ICF Operations and Functions

Figure 6-4 lists the CPI Communications calls and pseudonyms and the corresponding ICF operation or function, the DDS keyword, and the system-supplied format. The mappings provided

in this table apply when the CPI Communications conversations are at their default values. If a conversation characteristic is not a default value, one or more of these mappings may differ. For example, if SEND_TYPE is set to CM_SEND_AND_PREP_TO_RECEIVE, then the ICF operation or function would show a write operation; however, the DDS keyword would also include an ALWWRT.

Figure 6-4 (Page 1 of 2). Mapping of CPI Communications Calls to ICF Operations and Functions

Program Call	Pseudonym	ICF Operation or Function	DDS Keyword	System-Supplied Format
CMINIT	Initialize_Conversation	Open operation	Not applicable	Not applicable
CMALLC	Allocate	Combination of acquire operation and evoke function	EVOKE	\$\$EVOKNI
CMACCP	Accept_Conversation	Acquire to *REQUESTER	Not applicable	Not applicable
CMCFM	Confirm	Confirm function	CONFIRM	No equivalent
CMCFMD	Confirmed	Respond-to-Confirm function	RSPCONFIRM	No equivalent
CMDEAL	Deallocate	Write operation with detach function, and, depending on the program, one of the following: <ul style="list-style-type: none"> • For a source program, an end-of-session function, a release, or a close operation can follow the detach. • For a target program, an end-of-session function or close operation must follow the detach. 	DETACH EOS	\$\$SENDET \$\$EOS
CMFLUS	Flush	Force-data function	FRCDTA	No equivalent

Figure 6-4 (Page 2 of 2). Mapping of CPI Communications Calls to ICF Operations and Functions

Program Call	Pseudonym	ICF Operation or Function	DDS Keyword	System-Supplied Format
CMECT	Extract_Conversation_Type	Get-attributes operation	Not applicable	Not applicable
CMEMN	Extract_Mode_Name	Get-attributes operation	Not applicable	Not applicable
CMEPLN	Extract_Partner_LU_Name	Get-attributes operation	Not applicable	Not applicable
CMESL	Extract_Sync_Level	Get-attributes operation	Not applicable	Not applicable
CMPTR	Prepare_To_Receive	Allow-write or invite function Note: CPI Communications does not define an equivalent function to the implied LU 6.2 [MC_]POST_ON_RECEIPT that is done for an ICF invite function.	ALWWRT or INVITE	\$\$\$SEND with 0 length
CMRCV	Receive	Read operation to a specific program device	Not applicable	Not applicable
CMRTS	Request_To_Send	Request-to-write function	RQSWRT	\$\$RCD Note: This also causes an invite to be issued.
CMSEND	Send_Data	Write operation of a format with a data length greater than 0	Not applicable	\$\$\$SENDNI with data length greater than 0
CMSERR	Send_Error	Fail function	FAIL	\$\$FAIL
CMTRTS	Test_Request_To_Send_Received	No equivalent; however, the application program can check the request-to-write indicator in the I/O feedback area to determine if a Request_To_Send was received.	Not applicable	Not applicable

Example ICF Operations and Functions Mapped to CPI Communications Calls

Figure 6-5 provides some example ICF operations and functions that can be used in your application program and the CPI Communications call to which it corresponds. This table only provides DDS keyword examples and

should be used to supplement the more detailed information contained in Appendix D.

Notes:

1. Parameters that are required, such as the *conversation_ID* and *return_code*, are not always listed explicitly in the table.
2. These examples describe only one way of performing the operation or function; other methods may exist.

Figure 6-5 (Page 1 of 2). Mappings for Example ICF Operations and Functions

ICF Operations/Functions	CPI Communications Call
<p>An acquire operation followed by: EVOKE([library-name/]program-name)</p> <p>Note: It is the combination of an acquire operation and evoke function that corresponds to Allocate (CMALLC). With an evoke function, you can also specify any of the following: [parameter-1...[parameter-255]]</p> <p>SECURITY(3 *USER) or SECURITY([1 profile ID] [2 password] [3 userid])</p> <p>SYNLVL(*NONE)</p> <p>SYNLVL(*CONFIRM) : Write (data length > 0) with CONFIRM</p>	<p>Corresponds to CALL CMALLC(conversation_ID)</p> <p>CPI Communications does not support sending PIP data.</p> <p>CPI Communications only defines SECURITY(*SAME)</p> <p>CALL CMSSL(CM_NONE) (Must be issued before the CMALLC call)</p> <p>CALL CMSSL(CM_CONFIRM) (Must be issued before the CMALLC call)</p> <p>CALL CMSST(CM_SEND_AND_CONFIRM) followed by CALL CMSEND</p>
<p>If your program specifies SYNLVL(*CONFIRM) and later uses the CONFIRM DDS keyword, the remote program must then issue a</p> <p>RSPCONFIRM (positive response) or FAIL (negative response)</p> <p>If the remote program issues a Get-attributes operation, it can determine the SYNLVL established by the source program.</p>	<p>Corresponds to</p> <p>CALL CMCFMD or CALL CMSERR or CALL CMSED followed by CALL CMSERR when in send-pending state</p> <p>CALL CMESL</p>
<p>Your program issues a FRCDTA</p>	<p>Corresponds to CALL CMFLUS</p>
<p>Your program uses a Write operation</p> <p>to send data to the remote system, and then uses a Read operation</p> <p>to receive data from a specific program device (for example, PGMDEVA) and waits to receive data before continuing.</p>	<p>Corresponds to CALL CMSEND</p> <p>CALL CMRCV</p>
<p>If your program issues an: INVITE</p> <p>requesting data from the remote program to send data for a program device (for example, PGMDEVA), your program receives control without waiting for input. Later, your program can issue a Read-from-invited-program-devices operation</p> <p>A successful completion return code (set in the I/O feedback area) tells your program that data has been received and is in your program's buffer.</p>	<p>Corresponds to CALL CMSPTR(CM_PREP_TO_RECEIVE_FLUSH) followed by CALL CMPTR</p> <p>Note: CPI Communications does not support the implied LU 6.2 [MC]_POST_ON_RECEIPT that is done for the ICF invite function.</p> <p>No CPI Communications equivalent.</p> <p>However, you can issue a CMRCV with the <i>receive_type</i> set to CM_RECEIVE_IMMEDIATE for each conversation that is already in receive state. When this value is used, control is returned to your program immediately regardless of whether data has been received.</p>

Figure 6-5 (Page 2 of 2). Mappings for Example ICF Operations and Functions

ICF Operations/Functions	CPI Communications Call
<p>Your program issues: RQSWRT</p> <p>to indicate that it wants to send data to the remote program rather than continue receiving data.</p>	<p>Corresponds to CALL CMRTS</p>
<p>Your program issues a Write with ALWWRT</p> <p>to inform the remote program that your program is done sending data and is ready to receive.</p>	<p>Corresponds to CALL CMSPTR(CM_PREP_TO_RECEIVE_FLUSH) or CALL CMSST(CM_SEND_AND_PREP_TO_RECEIVE) and CALL CMSEND</p> <p>if data is sent with a change-direction indicator</p>
<p>Your program issues a Write with ALWWRT</p> <p>to inform the remote program that your program is done sending data and is ready to receive.</p>	<p>Corresponds to CALL CMSPTR(CM_PREP_TO_RECEIVE_FLUSH) and CALL CMPTR</p> <p>if no data is sent</p>
<p>To end a session, a source program issues: Write with DETACH, followed by EOS or close</p> <p>Note: The combination of DETACH and EOS corresponds to Deallocate (CMDEAL). The detach function indicates that the record is the last record to be sent and causes an APPC CEB to be sent.</p>	<p>Corresponds to CALL CMSDT(CM_DEALLOCATE_FLUSH) CALL CMSST(CM_SEND_AND_DEALLOCATE), followed by CALL CMSEND</p> <p>or CALL CMSDT(CM_DEALLOCATE_FLUSH) followed by CALL CMDEAL</p>
<p>If your program issues: DETACH with CONFIRM, followed by EOS or close,</p> <p>the transaction does not end if the remote program sends a FAIL (negative acknowledgement)</p> <p>Note: The fail function causes an error code and an APPC FMH7 to be sent to the remote system. Return codes are set in the file dependent area of the I/O feedback area.</p>	<p>Corresponds to CALL CMSDT(CM_DEALLOCATE_CONFIRM) and CALL CMDEAL</p> <p>CALL CMSERR</p> <p>Note: The SEND_ERROR call causes the Deallocate call to receive an error <i>return_code</i>.</p>
<p>The transaction ends if the remote program issues a RSPCONFIRM</p>	<p>Corresponds to CALL CMCFMD</p>
<p>If the program is a target program, started by a program start request, and it receives a detach indication using: RCVDETACH</p> <p>(or it receives return codes 0008, 0208, or 0308), it should end the session using: EOS or close</p>	<p>Corresponds to</p> <p>A return code of CM_DEALLOCATED_NORMAL</p> <p>Not applicable</p>
<p>If your program uses an EOS</p> <p>during an active transaction, both the transaction and the conversation are ended abnormally by APPC.</p>	<p>Corresponds to CALL CMSDT(CM_DEALLOCATE_ABEND) followed by CALL CMDEAL</p>

Flow Diagram for Inquiry Applications Using CPI Communications Calls

This flow parallels the one for ICF inquiry applications on page 5-19.

The steps shown in Figure 6-7 are described in Figure 6-6 :

Figure 6-7 on page 6-13 shows the flow for the exchange of information that occurs in an inquiry application using CPI Communications calls.

Figure 6-6 (Page 1 of 2). Description of Flow Diagram

Step	Description
1	<p>To communicate with its partner program, Program A must first establish a conversation. Program A uses the Initialize_Conversation call to tell CPI Communications that it wants to do the following:</p> <ul style="list-style-type: none"> • Initialize a conversation. • Identify the name of the target program (using <i>sym_dest_name</i>). • Ask CPI Communications to establish the identifier that the program uses when referring to the conversation (the <i>conversation_ID</i>). <p>On successful completion of the Initialize_Conversation call, CPI Communications provides the <i>conversation_ID</i> and returns it to Program A. The program must store the <i>conversation_ID</i> and use it on all subsequent calls intended for that conversation.</p>
2	<p>No errors occurred on the Initialize_Conversation call, so the <i>return_code</i> is set to CM_OK. Two major tasks are now accomplished:</p> <ul style="list-style-type: none"> • CPI Communications has established a set of conversation characteristics for the conversation, based on the <i>sym_dest_name</i>, and uniquely associates them with the <i>conversation_ID</i>. • Default values for the conversation characteristics are established.
3 and 4	<p>Program A sets the <i>sync_level</i> characteristic to CM_CONFIRM by issuing the Set_Sync_Level call.</p> <p>Note: Program A must set the <i>sync_level</i> characteristic before issuing the Allocate call (Step 9) if the new value is to take effect. Changing the <i>sync_level</i> after the conversation is allocated results in an error condition.</p>
5 and 6	<p>Program A sets the <i>send_type</i> characteristic to CM_SEND_AND_PREP_TO_RECEIVE by issuing the Set_Send_Type call. Setting the <i>send_type</i> to CM_SEND_AND_PREP_TO_RECEIVE means that Program A will be in Receive state after issuing a Send_Data call.</p>
7 and 8	<p>Program A sets the <i>deallocate_type</i> characteristic to CM_DEALLOCATE_FLUSH by issuing the Set_Deallocate_Type call.</p> <p>Note: The default value for the <i>deallocate_type</i> conversation characteristic is CM_DEALLOCATE_SYNC_LEVEL. A Deallocate call issued with a <i>sync_level</i> of CM_CONFIRM would start the Confirm call and if successful, deallocate the conversation normally.</p>
9 and 10	<p>Program A issues the Allocate call by using the <i>conversation_ID</i> to start the conversation. If a session between the logical units (LUs) is not already available, one is activated. Also note that the conversation start-up request is part of the Allocate processing.</p> <p>A <i>return_code</i> of CM_OK indicates that the Allocate call was successful and that the LU has allocated the necessary resources to the program for its conversation. Program A is now in Send state and can begin sending data.</p>
11	<p>Program A sends data (the part number) with the Send_Data call.</p> <p>Note: The Send_Data call is issued with the following conversation characteristics: a <i>send_type</i> of CM_SEND_AND_PREP_TO_RECEIVE; a <i>prepare_to_receive_type</i> of CM_PREP_TO_RECEIVE_SYNC_LEVEL; and a <i>sync_level</i> of CM_CONFIRM. Setting the conversation characteristics to these values flushes the data, changes the data flow direction, and sends a confirmation request to the partner program.</p>

Figure 6-6 (Page 2 of 2). Description of Flow Diagram

Step	Description
12 and 13	Program C responds by issuing a call to <code>Accept_Conversation</code> , which places the conversation into Receive state. The <code>Accept_Conversation</code> call is similar to the <code>Initialize_Conversation</code> call because it equates a <i>conversation_ID</i> with a set of conversation characteristics. Program C, like Program A in Step 2 , receives a unique <i>conversation_ID</i> that it will use in all future CPI Communications calls for that particular conversation. Some of the default characteristic values for Program C are based on the information contained in the conversation start-up request.
14 and 15	Program C sets the <i>send_type</i> characteristic to <code>CM_SEND_AND_PREP_TO_RECEIVE</code> by issuing the <code>Set_Send_Type</code> call.
16 and 17	Program C sets the <i>prepare_to_receive_type</i> conversation characteristic to <code>CM_PREP_TO_RECEIVE_FLUSH</code> by issuing the <code>Set_Prepare_To_Receive_Type</code> call.
18	Program C issues the <code>Receive</code> call to receive the data.
19	A <i>return_code</i> of <code>CM_OK</code> indicates successful receipt of data. Program C also receives a request for confirmation, indicated by the <i>status_received</i> parameter, which is set to <code>CM_CONFIRM_SEND_RECEIVED</code> . Program C is in Confirm-Send state and must respond to the confirmation request.
20 and 21	Program C responds to the confirmation request by issuing a <code>Confirmed</code> call, which completes the program as send data with a <i>return_code</i> of <code>CM_OK</code> .
22	Program A issues a <code>Receive</code> call to receive the data.
23	Program C sends the data that was requested using the <code>Send_Data</code> call, and is now in Receive state because the <i>send_type</i> was set to <code>CM_SEND_AND_PREP_TO_RECEIVE</code> . The data and the permission-to-send indication are transmitted to Program A. The <code>Receive</code> call issued by Program A completes successfully. Program A received data with the permission-to-send indication. Program A is in Send-Pending state and can send data to the partner program.
24	Program C uses the <code>Receive</code> call to receive the data.
25	Program A issues a <code>Deallocate</code> call to release the conversation. Because the <i>deallocate_type</i> characteristic was set to <code>CM_DEALLOCATE_FLUSH</code> (see Step 7), the local data buffer of the LU is flushed and the conversation is deallocated normally. The partner program (Program C) receives the deallocate notification by means of a <i>return_code</i> of <code>CM_DEALLOCATED_NORMAL</code> .
26	A return code of <code>CM_OK</code> tells Program A that the conversation is successfully deallocated. Both Program A and Program C complete normally.

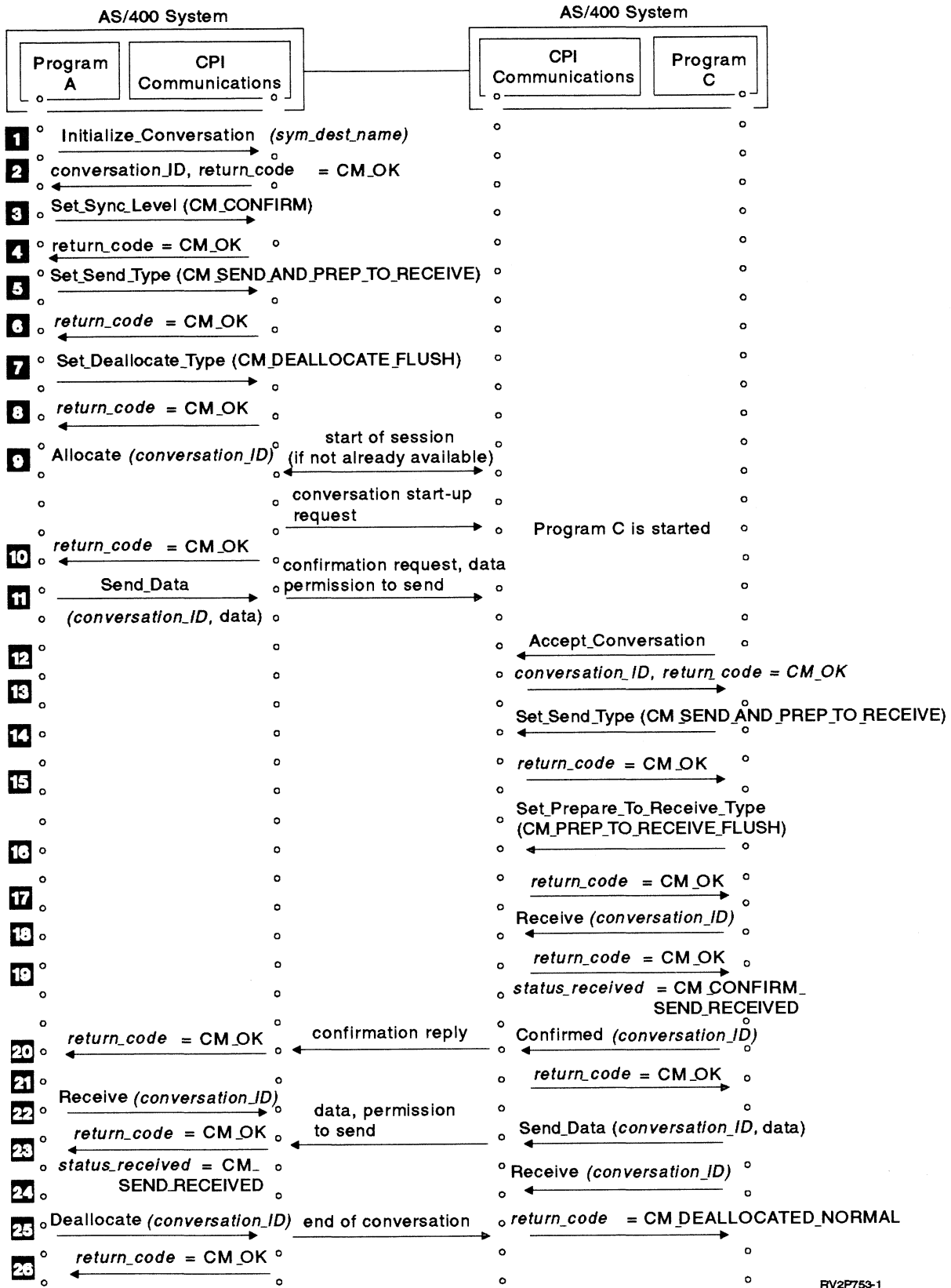


Figure 6-7. Data Flow Using CPI Communications Calls

Return Codes for CPI Communications

The return codes for CPI Communications form a one-to-one correspondence with the LU 6.2 architected return codes. Figure 6-8 on page 6-14 provides a return code mapping

between CPI Communications and the LU 6.2 architecture. The *CPI Communications Reference* contains detailed explanations of the return codes.

Figure 6-8. Return Code Mapping for CPI Communications and LU 6.2

LU 6.2 Return Code	CPI Communications Return Code
ALLOCATION_FAILURE_NO_RETRY	CM_ALLOCATE_FAILURE_NO_RETRY
ALLOCATION_FAILURE_RETRY	CM_ALLOCATE_FAILURE_RETRY
CONVERSATION_TYPE_MISMATCH	CM_CONVERSATION_TYPE_MISMATCH
DEALLOCATE_ABEND	CM_DEALLOCATED_ABEND
DEALLOCATE_ABEND_PROG	CM_DEALLOCATED_ABEND_PROG
DEALLOCATE_ABEND_SVC	CM_DEALLOCATED_ABEND_SVC
DEALLOCATE_ABEND_TIMER	CM_DEALLOCATED_ABEND_TIMER
DEALLOCATE_NORMAL	CM_DEALLOCATED_NORMAL
OK	CM_OK
PARAMETER_ERROR	CM_PARAMETER_ERROR
PIP_NOT_SPECIFIED_CORRECTLY	CM_PIP_NOT_SPECIFIED_CORRECTLY
PROG_ERROR_NO_TRUNC	CM_PROGRAM_ERROR_NO_TRUNC
PROG_ERROR_PURGING	CM_PROGRAM_ERROR_PURGING
PROG_ERROR_TRUNC	CM_PROGRAM_ERROR_TRUNC
PROGRAM_PARAMETER_CHECK	CM_PROGRAM_PARAMETER_CHECK
PROGRAM_STATE_CHECK	CM_PROGRAM_STATE_CHECK
RESOURCE_FAILURE_NO_RETRY	CM_RESOURCE_FAILURE_NO_RETRY
RESOURCE_FAILURE_RETRY	CM_RESOURCE_FAILURE_RETRY
SECURITY_NOT_VALID	CM_SECURITY_NOT_VALID
SYNC_LEVEL_NOT_SUPPORTED_BY_LU	CM_SYNC_LVL_NOT_SUPPORTED_LU
SYNC_LEVEL_NOT_SUPPORTED_BY_PGM	CM_SYNC_LVL_NOT_SUPPORTED_PGM
SVC_ERROR_NO_TRUNC	CM_SVC_ERROR_NO_TRUNC
SVC_ERROR_PURGING	CM_SVC_ERROR_PURGING
SVC_ERROR_TRUNC	CM_SVC_ERROR_TRUNC
TPN_NOT_RECOGNIZED	CM_TPN_NOT_RECOGNIZED
TP_NOT_AVAIL_NO_RETRY	CM_TP_NOT_AVAILABLE_NO_RETRY
TP_NOT_AVAIL_RETRY	CM_TP_NOT_AVAILABLE_RETRY
UNSUCCESSFUL	CM_UNSUCCESSFUL
Not applicable	CM_PRODUCT_SPECIFIC_ERROR

Chapter 7. Application Considerations for ICF

This chapter describes the application considerations for the ICF interface.

Before writing programs that use APPC, you should understand some of the characteristics of the AS/400 environment.

General Considerations

- The first operation following the open or acquire operation by a source program should be a write operation with an evoke function specified. This causes the target program, with which the source program is going to communicate, to be started.
- The source program can send program initialization parameters (PIP) to the target program only if the remote system or program permits PIP data to be received.
- Target programs on the AS/400 system establish a connection to the session and transaction started by the source program by issuing an acquire operation to the program device associated with the remote location name *REQUESTER (the requesting program device).
- The first operation following the open and acquire of the requesting program device by a target program can be a read operation. The target program always starts in the receive state.
- When a program is in the receive state, it can issue a read operation, an invite function, or a request-to-write function. A write operation issued with a fail function may also be used if your program is to send an error condition to the remote system.
- When a program is in the send state, all operations except open (to the same ICF file), acquire (to the same program device), and the request-to-write function are valid.
- When a program is in the receive state, it must continue to issue input operations until one of the following is received:
 - A minor return code indicating that your program may now send. The RCVTRNRND keyword may also be used to indicate that your program may send.
 - A minor return code indicating that detach has been received. The RCVDETACH keyword may also be used.
 - A major and minor return code indicating an error condition. For example, any of the 80 or 81 major return codes.
- If a target program never opens and acquires the requesting program device, a diagnostic message, CPF4059, is written to the job log when the target program completes. The message is normal if there is no need for the target program to communicate with the source program. If the target program is supposed to communicate with the source program, then this message indicates a possible logic problem in your program.

Open and Acquire Operation Considerations

- If a target program acquires (explicitly or implicitly) a program device other than the requesting program device, a new session is established and the connection with the source program is not established. No error is indicated because it is valid for a target program on one session to be a source program on another session. If the program issues an input operation as the first operation to the newly established session, it will receive a return code indicating that no transaction is active.
- The conversation type specified on the program device entry used by the target program can be the same as the conversation type specified for the source program, or it can be *SRCPGM. If the conversation types do not match explicitly, the open or acquire will fail.
- When a session is established by a source program, the CPI9803 message is sent to the history file. This message is also sent to the history file when a target program is started on an AS/400 system. The CPI9803 message contains the unit-of-work ID for the job.
- Multiple sessions that run at the same time can be established with another system, pro-

viding that the other system supports multiple sessions. The program device names are used to distinguish the sessions within your program.

- The program uses the location parameters on the ADDICFDEVE or OVRICFDEVE commands to tell the system to which remote location the session is to be established. Refer to "Using the Location Parameters," for information about specifying location information.

WAITFILE Considerations

- The WAITFILE parameter of the create, change, or override ICF file command determines how long the APPC support will wait for session resources to become available when performing an acquire operation or an evoke function. If the WAITFILE timer runs out, a return code of either 82B3 or 81C2 is returned. The WAITFILE value is not used for sessions where the connection to the adjacent system is over a switched connection, for example, an SDLC switched line, an X.25 SVC, Ethernet, or a token-ring connection. Acquire operations and evoke functions do not time out when using switched connections that the acquire or evoke causes to be dialed. If the line is already dialed, the WAITFILE parameter is used. However, an evoke function following a successful detach function that was already sent or received will fail if it is sent on a disconnected switched line.

Note: This is only true for devices specified APPC(*NO). For devices specified APPC(*YES), the WAITFILE parameter is always used.

- Because APPN sessions may cross multiple systems and lines to reach the remote system, the WAITFILE timer should be adjusted to allow more time in these cases. You should not specify *IMMED for the WAITFILE parameter if your application is running in a network configured to use APPN functions. The value you specify for this parameter is dependent on the size and type of the network.

Output Considerations

- Handling write operations with a 0-length record: if ALWWRT, CONFIRM, DETACH, EVOKE, or FAIL is used with a 0-length record, APPC assumes that the functions to be performed are only those indicated by the keywords and the record is not placed in the transmission block.
- Because APPC support can have up to 16K of data in its buffer before sending it to the remote system, the request-to-write or fail indicators from the remote system may be delayed. Therefore, you need to design your application program so that the time delay between sending, receiving, and responding to these indicators is kept to a minimum. You can do this by using the force-data, confirm, invite, or allow-write function. These functions allow you to force APPC to flush all buffered data and make it available to the partner program.
- The force-data function causes any data currently held in the buffer to be sent immediately to the remote system. Because APPC support does not wait until its buffer is full (there is 16KB of data in its buffer) before sending the data, it takes more processing unit time to transfer a large amount of data to the remote system than if the system waited for each data buffer to be full. Therefore, if you use the force-data function on each write operation, you should be aware that you may experience slower response times because the data is less efficiently transferred.
- You may want to send certain data as control data. The Control-Data (CTLDTA) keyword allows you to make this distinction. For more information about CTLDTA, see "Sending Data" on page 5-6.

Input Considerations

- The receive indicators (RCVTRNRND, RCVDETACH, RCVCTLDTA, and RCVCONFIRM) can be received either with data or without data (indicators only). Your program should examine the major return codes in the communications device-dependent I/O feedback area to determine if the record contains data. A major code of 00 or 02 indicates data has been received, and

a major code of 03 indicates no data has been received.

- When the actual length of the data received is less than the length of the record format used to receive the data and mapped conversation is being used, the system pads the remainder of the record with blanks. The actual received data length can be determined from the I/O feedback area.
- The value of the ICF program device entry Overflow Data (OVRFLWDTA) parameter is used to determine whether to retain or discard the remainder of the data in these cases:
 - When the actual length of the data received is greater than the length of the record format used to receive the data and mapped conversation is being used.
 - Basic conversation is being used without the VARBUFMGMT keyword.

For more information on the overflow data function, see OVRFLWDTA on page 5-3.

- When a read operation is issued in send state or a turnaround indication was received as a result of issuing the read operation, the system performs the equivalent of an allow-write function and begins waiting for data from the remote program.

Confirm Considerations

- Your program requests that the remote program confirm receiving the data by issuing an output request with the confirm function.
- Your program is notified that it has received a confirmation request from the remote program in the following ways:
 - A major return code of 00, 02, or 03, with minor return codes of 13, 14, 15, 18, 1C, 1D, 44, 45, or 46,
 - The RCVCONFIRM indicator is set.

Once your program has received a confirmation request, it must either respond positively or negatively to the request as follows:

- To respond positively, issue the RSPCONFIRM function.
- To respond negatively to the request, do the following:

- Issue a fail function. In this case, your program is responsible for initiating the appropriate level of error recovery.
- Abnormally end the transaction and session by issuing either an end-of-session function or a close operation.

- If an evoke function is issued with a confirmation request, and the target program never attaches to the transaction, a negative response is returned to the source program when the target program completes.

Do not use the confirm function with an evoke function to start a target program that does not read data from or write data to the source program. If a confirmation is required by your application program, the target program must receive the confirmation request and perform the necessary function to respond to the confirm request.

- When it is essential to your application program that the target program be started before you issue output operations to it, specify the confirm function with the evoke function. The evoke will not complete until the target program responds to the confirmation request.
- When it is not essential to your application program that the target program be started before you issue output operations to it, issue the evoke function without a confirm function specified. This allows your program to issue subsequent output operations. These output operations are processed during the program start processing at the target system. If a problem occurs when starting the target program, the source program is not informed of the problem until a later receive or send operation.
- Because the output operation with the confirm function specified waits for a positive or negative response before control is returned to the program, it is recommended that corresponding source and target programs be coded to minimize the amount of time between receiving the confirm request and sending the response. If the receiving program performs complex processing on each record, the time delay can be significant.

- The confirm function causes any data currently held in the buffer to be sent immediately to the remote system. Because APPC support does not wait until the buffer is full before sending the data, it takes more time to transfer data to the remote system than if the system waited for the data buffer to be full. Therefore, if you use the confirm function, you should be aware that you may experience slower response times because data is transferred less efficiently, and the program must wait for a response from the remote program.

End-of-Session, Release, and Close Considerations

The following information describes how the end-of-session function and the close and release operations are used to end communications between your local program and the target program.

- The end-of-session function is valid in any state, but will abnormally end an active transaction with the remote program and could also indicate a logic error in the program.
- A target program cannot initiate error recovery using close, release, and open and acquire logic. When a permanent session error occurs, the source program is responsible for recovery.
- For source programs, when you issue a close operation to a session with an active transaction while your program is in a send state, the system sends any data in the output buffer and then abnormally ends the transaction.
- A release operation performed by the target program does not perform a detach function. The transaction with the source program may be resumed by a subsequent acquire of the requesting program device. That acquire may be performed either by the program that initially had the transaction or by another program running in the same job.
- A transaction remains allocated to a target job until the job ends even though a close or release operation was issued and a detach sent. As long as the job is active, the WRKCFGSTS commands will show the job as an APPC target program. You can use the end-of-session function to end the session associated with a job. In this case, the job is no longer shown as active on the WRKCFGSTS commands.

Prestart Jobs Considerations

To minimize the amount of time required to carry out a program start request, you can use prestart jobs to start a job on your system before the remote system sends a program start request.

To use prestart jobs, you will need to define both communications and prestart job entries in the subsystem description, and make certain programming changes to the prestart job program with which the source program communicates. You should make as many changes as possible before acquiring the program with the RMTLOCNAME(*REQUESTER) parameter specified on the program device entry. For information about how to use prestart job entries, refer to the *ICF Programmer's Guide*. For information on the prestart job entry format, refer to the *Work Management Guide*.

Trace ICF Communications Considerations

The **Trace ICF** (TRCICF) command enables you to trace ICF calls and data passed on those calls and directs the information obtained to either a printer file or an output file. You can use the TRCICF command for remote jobs by using it in conjunction with the Start Service Job (STRSRVJOB), End Service Job (ENDSRVJOB), and Display Service Job (DSPSRVJOB) commands.

For more information about the Trace ICF command, refer to the *ICF Programmer's Guide*.

Chapter 8. Application Considerations for CPI Communications

This chapter describes the application considerations for the CPI Communications interface.

Before writing programs that use CPI Communications and APPC, you should understand some of the characteristics of the AS/400 environment.

General Considerations

- When a source program issues an Allocate (CMALLC) call, a conversation start request (program start request) is sent to the target system for the partner program.
- When an Allocate (CMALLC) call is issued and the conversation characteristic *return_control* is CM_IMMEDIATE, control is returned immediately if an active, locally controlled session is not available. This implies special considerations when an APPN *partner_LU_name* is requested. Because the request is for an available session, APPN support will not dynamically create or vary on the device, attach the required *mode_name*, or issue the start mode (STRMOD) command for you. You must use a *return_control* of CM_WHEN_SESSION_ALLOCATED to take advantage of these APPN services.
- A target program accepts an incoming conversation by issuing an Accept_Conversation (CMACCP) call.
- When an Allocate or Accept_Conversation call completes successfully, the message CPI9803 is sent to the history file. The CPI9803 message contains the unit-of-work ID for the job. The unit-of-work ID can be used to correlate which jobs are communicating with each other when the local and target systems are AS/400 systems.
- Multiple conversations that run simultaneously can be established within one source job. The *conversation_ID* is used to distinguish the conversations in your program.
- A target program can also establish multiple conversations. A target program can have only one *incoming* conversation, which is established by issuing the Accept_Conversation (CMACCP) call. A

target program can also establish additional conversations by issuing the Initialize_Conversation (CMINIT) and Allocate (CMALLC) calls.

- Once the incoming conversation has been deallocated, a target program that is a prestart job can issue another Accept_Conversation (CMACCP) call to wait for and process another incoming conversation. Target programs that are not prestart jobs can never accept another incoming conversation, even when the first incoming conversation is deallocated. See "Prestart Job Considerations" on page 8-2 for more information.

Performance Considerations

- APPC can buffer up to 16KB of data internally before sending the data to the partner program. This allows for a more efficient transmission of data on the AS/400 system. The data is sent to the partner system when it becomes full, or when a CPI Communications call is issued that forces APPC to flush the buffer. The calls that will cause APPC to flush the buffer are
 - Confirm
 - Deallocate
 - Flush
 - Prepare_to_Receive
 - Receive
 - Send_Data when the *send_type* is anything other than CM_BUFFER_DATA
- As a general rule, using a larger buffer size improves performance. For example, a program that sends 10,000 records that are 100 bytes long will not perform as well as a program that sends 1,000 records that are 1,000 bytes long, even though it is the same amount of data. You may consider changing the programs to use a *conversation_type* of CM_BASIC_CONVERSATION and a *fill* of CM_FILL_BUFFER. Using this approach, your program can send multiple records with each Send_Data call, which can improve performance.
- Special consideration should be given to the frequency with which your program issues

any call that causes the partner program to receive a *status_received* value of `CM_CONFIRM_RECEIVED`, `CM_CONFIRM_SEND_RECEIVED`, or `CM_CONFIRM_DEALLOC_RECEIVED`. Your program does not receive control until the partner program issues an operation to respond to this call, such as `Confirmed`, and the time delay may be significant. When performance is important, calls that require confirmation should be used only when necessary.

- The time required to start a job to process an incoming conversation can be significant. Refer to “Prestart Job Considerations” for information on reducing this time delay.

Prestart Job Considerations

To minimize the time required for your program to accept a conversation with its partner program, you can use a prestart job entry. When you use a prestart job entry, your program is started before a program start request is received from the partner program. Each prestart job entry contains a program name, library name, user profile, and other attributes that the subsystem uses to create and manage a pool of prestart jobs.

To use a prestart job entry, you must

- Define a prestart job entry in the subsystem that contains the communications entry using the Add Prestart Job Entry (ADDPJE) command.
- Start the prestart job entry. The prestart job entry can be started at the same time the subsystem is started, or you can use the Start Prestart Jobs (STRPJ) command.

You should consider the following when writing a program that uses a prestart job entry:

- A prestart job program should do as much work as possible before issuing the `Accept_Conversation` call (for example,

opening database files). This allows the initial processing to be completed before the program start request is received. The `Accept_Conversation` call will not complete until a program start request is received for your program. When the program start request is received, your program receives control with a *return_code* of `CM_OK` (assuming no authorization or other problems are encountered), and your program can immediately begin processing.

- When a prestart job program is done servicing a program start request (a *return_code* of `CM_DEALLOCATED_NORMAL` is received or a call to `Deallocate` completes successfully), it may then make itself available for another program start request by issuing another `Accept_Conversation` call.
- Only resources that are specifically used for a conversation should be deallocated. For example, if a database file is used for most conversations, there is no need to close the file and then open it each time a conversation is deallocated and a new conversation is accepted.

See the *Work Management Guide* for more information on prestart job entries.

Trace CPI Communications Considerations

The **Trace CPI Communications** (TRCCPIC) command enables you to trace CPI Communications calls and data passed on those calls and directs the information obtained to either a printer file or an output file. You can use the TRCCPIC command for remote jobs by using it in conjunction with the Start Service Job (`STRSRVJOB`), End Service Job (`ENDSRVJOB`), and Display Service Job (`DSPSRVJOB`) commands.

For more information about the Trace CPI Communications command, refer to the *Communications Management Guide*.

Appendix A. ICF Language Operations, Data Description Specifications Keywords, and System-Supplied Formats

Information in this appendix includes the following for APPC:

- All valid language operations supported by ICF
- Valid operations for each programming language that supports ICF
- Data description specifications (DDS) processing keywords
- System-supplied formats

Language Operations

Figure A-1 describes the language operations supported by ICF.

Figure A-1. Language Operations

ICF Operations	Description
Open	Opens the ICF file.
Acquire	Establishes a session between the application and the remote location.
Get attributes	Used to determine the status of the session.
Read	Obtains data from a specific session.
Read-from-invited-program-devices	Obtains data from any session that has responded to an invite function.
Write	Passes data records from the issuing program to the other program in the transaction.
Write/Read	Allows a write operation followed by a read operation.
Release	Attempts to end a session.
Close	Closes the ICF file.

Language Operations Supported

Use high-level language operations and ICF to communicate with a program at a remote system. (See the specific high-level language manual for the non-ICF operations.)

Figure A-2 presents the ICF file operations used with APPC communications and the equivalent high-level language statement.

Figure A-2. High-Level Language I/O Operations

ICF Operation	RPG/400 Operation Code	COBOL/400 Procedure Statement	C/400 Function ¹	FORTRAN/400 Statement
Open	OPEN	OPEN	fopen or _Ropen	OPEN
Acquire	ACQ	ACQUIRE	QXXACQUIRE or _Racquire	Not supported ²
Get attributes	POST	ACCEPT	QXXDEVATR or _Rdevatr	Not supported
Read	READ	READ	fread or _Rreadn	READ
Read-from-invited-program-devices	READ ³	READ ³	QXXREADINVDEV followed by an fread or _Rreadindv	Not supported
Write	WRITE	WRITE	fwrite or _Rwrite	WRITE
Write/Read	EXFMT	Not supported	_Rwriterd	Not supported
Release	REL	DROP	QXXRELEASE or _Rrelease	Not supported
Close	CLOSE	CLOSE	fclose or _Rclose	CLOSE

Notes:

- ¹ C/400 is case sensitive.
- ² FORTRAN/400 does not support multiple sessions for one ICF file. Therefore, program device names are not used in a FORTRAN/400 program. An implicit acquire must be used by specifying ACQPGMDEV (program device name) on the CRTICFF, CHGICFF, or OVRICFF commands.
- ³ A read operation can be directed either to a specific program device or to any invited program device. The support provided by the compiler you are using determines whether to issue an ICF read or read-from-invited-program-devices operation, based on the format of the read operation. For example, if a read is issued with a specific format or terminal specified, the read operation is interpreted as an ICF read operation. Refer to the appropriate language reference manual for more information.

Data Description Specifications Keywords

Figure A-3 lists the DDS keywords that are valid for APPC.

Figure A-3 (Page 1 of 2). DDS Keywords

DDS Keyword	Description
ALWWRT	The record currently being written ends a transmission. The program goes to receive state.
CONFIRM	Requests that the remote program confirm receipt of data.
CTLDTA	Causes the data to be sent as control data.
DETACH	Informs the remote program that the sending program is ending the transaction.

Figure A-3 (Page 1 of 2). DDS Keywords

DDS Keyword	Description
DFREVOKE	Delays an evoke until the send buffer is full or an ALWWRT, CONFIRM, DETACH, FRCDTA, INVITE, or READ function is issued.
EOS	Used to specify an end-of-session function.
EVOKE	Starts a program on the remote system.
FAIL	Sends a fail indicator to the remote system.
FMTNAME	Specifies that a format-name should be sent on output operations.
FRCDTA	Immediately sends communications data currently in the buffer without waiting for the buffer to become full.

Figure A-3 (Page 2 of 2). DDS Keywords

DDS Keyword	Description
INVITE	Schedules an invite function.
RCVCONFIRM	Indicates that the remote program is requesting a confirmation of transaction activity.
RCVCTLDTA	Informs the local program that control data has been received.
RCVDETACH	Indicates that the remote program has ended the transaction.
RCVFAIL	Indicates that the remote program has issued a fail indicator.
RCVTRNRND	Indicates that the program is now in send state.
RECID	Used to allow the data content to identify the record format to use to receive the data.
RQSWRT	Specifies that the program is requesting permission to write.
RSPCONFIRM	Used to respond positively to a received confirm request.
SECURITY	Includes security information needed to start a program on the remote system.
SYNLVL	Includes the synchronization level of the program.
TIMER	Allows the user to specify an interval of time to wait before a read-from-invited-program-devices operation receives a timer-expired return code.
VARBUFMGMT	Allows the user to send or receive multiple or partial records using one record format for each read or write operation.
VARLEN	Specifies that the length of the user data to be sent will be specified in the 5 bytes of the field specified.

System-Supplied Formats

Figure A-4 lists all the keyword functions performed by the system-supplied formats that are valid for APPC. Refer to *ICF Programmer's Guide* for more information about system-supplied formats.

Figure A-4. System-Supplied Formats

System-Supplied Formats	Description
\$\$EOS	End-of-session
\$\$EVOK	Evoke with invite
\$\$EVOKET	Evoke with detach
\$\$EVOKNI	Evoke
\$\$FAIL	Fail
\$\$RCD	Request-write-with-invite
\$\$SEND	Send-then-invite or invite
\$\$SENDET	Send-then-detach or detach
\$\$SENDNI	Send
\$\$TIMER	Timer

Appendix B. Sense Data and Return Codes

Program Start Request Errors

When a program start request is rejected by the system, message CPF1269 is sent to the system operator message queue. This message contains information that can be used to determine why the program start request was rejected.

The SNA FMH7 sense data shown in Figure B-1 is sent to the remote system (the one that issued the program start request). The reason code causes the associated message text to be sent to the local system operator message queue.

Note: FMH7 sense data will only be sent if the conversation is active and allows a response after the failure has been detected (that is, the source program issues a confirm or an input operation).

Figure B-1 (Page 1 of 4). Reason Codes for Rejected Program Start Requests

SNA FMH7 Sense Data in Hexadecimal	Reason Code	Reason Description
084B6031	401	Program start request received for a device that is not allocated to an active subsystem.
084B6031	403	User profile is not accessible.
084B6031	404	Job description is not accessible.
084B6031	405	Output queue is not accessible.
084B6031	406	Maximum number of jobs, defined by subsystem description, is already active.
084B6031	407	Maximum number of jobs, defined by communications entry, is already active.
084B6031	408	Maximum number of jobs, defined by routing entry, is already active.
084B6031	409	Library on library list is in use exclusively by another job.
084B6031	410	Group profile cannot be accessed.
084B6031	411	Insufficient storage in machine pool to start job.
084B6031	412	System job values not accessible.
084C0000	501	Job description is not found.
084C0000	502	Output queue is not found.
084C0000	503	Class is not found.
084C0000	504	Library on library list is not found.
084C0000	505	Job description or job description library is damaged.
084C0000	506	Library on library list is destroyed.
084C0000	507	Duplicate libraries are found on library list.
084C0000	508	Defined size of storage pool is zero.
10086021	602	Value of transaction program name is reserved but not supported.
10086021	604	Matching routing entry is not found.

Figure B-1 (Page 2 of 4). Reason Codes for Rejected Program Start Requests

SNA FMH7		
Sense Data in Hexadecimal	Reason Code	Reason Description
10086021	605	Program was not found in the library you specified, or the program was not found in the default library list (QSYSLIBL).
080F6051	704	Password is not valid.
080F6051	705	User is not authorized to device.
080F6051	706	User is not authorized to subsystem description.
080F6051	707	User is not authorized to job description.
080F6051	708	User is not authorized to output queue.
080F6051	709	User is not authorized to program.
080F6051	710	User is not authorized to class.
080F6051	711	User is not authorized to library on library list.
080F6051	712	User is not authorized to group profile.
080F6051	713	User ID is not valid.
080F6051	714	Default user profile is not valid.
080F6051	715	Neither password nor user ID is provided, and no default user profile is specified in the communications entry.
080F6051	722	A user ID was received without a password.
080F6051	723	There is no password associated with the system user ID.
080F6051	725	User ID is not a valid AS/400 name.
080F6051	726	User profile disabled.
10086031	801	Program initialization parameters are present but not allowed.
084B6031	802	Program initialization parameter exceeds 2000 bytes.
084B6031	803	Subsystem is ending.
084B6031	804	Prestart job is inactive or is ending.
084B6031	805	WAIT(*NO) was specified on the prestart job entry, and no prestart job was available.
084B6031	806	The maximum number of prestart jobs that can be active on a prestart job entry was exceeded.
084B6031	807	Prestart job ended when a program start request was being received.
10086032	901	Program initialization parameters are not valid.
10086032	902	Number of parameters for program not valid.
10086032	903	Program initialization parameters required but not present.
08640001	1001	System logic error; function check or unexpected return code encountered.
08640001	1002	System logic error; function check or unexpected return code encountered while receiving initialization parameters.
10086021	1501	Character as procedure name is not valid.
10086021	1502	Procedure not found.
084C0000	1503	S/36 environment library not found.
084C0000	1504	Library QSSP not found.

Figure B-1 (Page 3 of 4). Reason Codes for Rejected Program Start Requests

SNA FMH7		
Sense Data in Hexadecimal	Reason Code	Reason Description
084C0000	1505	File QS36PRC not found in library QSSP.
10086021	1506	S/36 Procedure or library name is greater than 8 characters.
084C0000	1507	Current library not found.
084C0000	1508	Not authorized to current library.
080F6051	1509	Not authorized to Q36PRC in current library.
080F6051	1510	Not authorized to procedure in current library.
080F6051	1511	Not authorized to S/36 environment library.
080F6051	1512	Not authorized to file QS36PRC in S/36 environment library.
080F6051	1513	Not authorized to procedure in S/36 environment.
080F6051	1514	Not authorized in library QSSP.
080F6051	1515	Not authorized to file QS36PRC in QSSP.
080F6051	1516	Not authorized to procedure in QS36PRC in QSSP.
08640001	1517	Unexpected return code from S/36 environment support.
10086021	1518	Problem phase program not found in QSSP.
080F6051	1519	Not authorized to problem phase program in QSSP.
084B6031	1520	Maximum number of target programs started (100 for S/36 environment).
10086000	2001	FMH5 length field is not correct.
1008200E	2002	Concatenation flag not valid.
10084001	2003	FMH type is not 5.
1008600B	2004	Command code field in FM header is not valid.
10086009	2005	Length for fixed-length fields not valid.
10086034	2006	Conversation type is not supported.
10086040	2007	Sync point level is not supported.
10086040	2008	Reconnection is not supported.
10086021	2009	Field length of transaction program name not valid.
10086005	2010	Access code subfield length not valid.
10086011	2011	UOW-ID subfield length not valid.
10086011	2012	UOW-ID contents not valid.
084B6031	2013	Requested device is currently being held by a Hold Communications Device (HLDCMNDEV) command.
10086021	2014	Transaction program name value is reserved but not supported.
10086021	2015	LU service request received but the LU service job is not active.
10086040	2016	SECURELOC(*NO) specified and program start request indicated remote system did verification (AVI indicator).
080F6051	2017	No user ID is provided, but a password is sent.
080F6051	2018	No user ID is provided, but a profile ID is sent.

Figure B-1 (Page 4 of 4). Reason Codes for Rejected Program Start Requests

SNA FMH7		
Sense Data in Hexadecimal	Reason Code	Reason Description
080F6051	2019	Remote system indicates it sent a verified user ID, but no user ID is provided.
080F6051	2020	Remote system sent a verified user ID, but also sent a password.
080F6051	2021	Remote system sent a user ID, which it had not verified, and failed to send a password.
080F6051	2022	Password received and system is a nonsecure system.

SNA Sense Data

For some ICF return codes, the negative-response data field in the input/output feedback area contains SNA sense data received from the remote system, indicating the reason for a negative-response.

Note: CPI Communications does not return sense data.

Refer to the *APPN Guide* and the *SNA Formats* manual for information about sense data.

Return Codes

This section describes all the return codes that are valid for APPC. These return codes are set in the I/O feedback area of the ICF file; they report the results of each I/O operation issued by your application program. Your program should check the return code and act accordingly. Refer to your high-level language manual for more information on how to access these return codes.

Each return code is a four-digit hexadecimal value. The first two digits contain the *major code*, and the last two digits contain the *minor code*.

With some return codes, a message is also sent to the job log or the system operator message queue (QSYSOPR). You can refer to the message for additional information.

Notes:

1. In the return code descriptions, *your program* refers to the local AS/400 application program that issues the operation and receives a return code from ICF communications. The *partner program* refers to the application program on the remote system with which your program is communicating through ICF.
2. Several references to input and output operations are made in the descriptions. These operations can include DDS keywords and system-supplied formats, which are listed in Appendix A.

Major Code 00

Major Code 00 – Operation completed successfully.

Description: The operation issued by your program completed successfully. Your program may have sent or received some data, or may have received a message from the remote system.

Action: Examine the minor return code and continue with the next operation.

Code Description/Action

0000 **Description:** For input operations issued by your program, 0000 indicates that your program received some data with a turnaround indication. The partner program is ready to receive data.

For output operations issued by your program, 0000 indicates that the last output operation completed successfully and that your program can continue to send data.

Action: If your program received a turnaround on an input operation, issue an output operation. For the actions which can be taken after 0000 is received, refer to Figure B-2:

Figure B-2. Actions for Return Code 0000

Type of Session	Last Operation Issued	Actions Your Program Can Take
Started by a source program	Acquire or open	Issue an evoke or timer function, or a get-attributes operation.
	Evoke with detach or write with detach	Issue another evoke function, issue a release operation, continue local processing, or end.
	Any other output operation	Issue another output operation (except evoke), or issue an input operation.
	End-of-Session	Continue local processing or end.
Started by a remote program start request ¹	Acquire or open	Issue an input or output operation.
	Write with detach	Continue local processing or end. This session has ended.
	Any other output operation	Issue another output operation (except evoke), or issue an input operation.
	End-of-Session	Continue local processing or end.

¹ A target program (started by a program start request) cannot issue an evoke function in this session; it can issue an evoke function only in a different session that it has first acquired.

0001 **Description:** On a successful input operation, your program received some data. Your program must continue to receive data until it receives a turnaround indication (which allows your program to send data) or a detach indication.

Action: Issue another input operation. If your program detects a turnaround indication, it can issue an output operation.

0002 **Description:** Your program received control data, sent with the program start request, on the first input operation. Your program must continue to receive data until it receives a turnaround indication (which allows your program to send data) or a detach indication.

Action: Issue another input operation. If your program detects a turnaround indication, it can issue an output operation.

0004 **Description:** On a successful input operation, your program received some control data with a turnaround indication. The partner program is ready to receive data.

0005 **Description:** On a successful input operation, your program received some control data. Your program must continue to receive data until it receives a turnaround indication (which allows your program to send data) or a detach indication.

Action: Issue another input operation. If your program detects a turnaround indication, it can issue an output operation.

0006 **Description:** Your program received control data, sent with the program start request, on the first input operation, along with a turnaround indication.

0008 **Description:** On a successful input operation, your program received a detach indication with the last of the data. The communications transaction with the partner program has ended, but the session with the partner system is still active.

Action: If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the partner program started the transaction, your program can either issue an end-of-session function or end.

000C **Description:** On a successful input operation, your program received a detach indication with the last of the control data. The communications transaction with the partner program has ended, but the session with the partner system is still active.

Action: If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the partner program started the transaction, your program can either issue an end-of-session function or end.

0010 **Description:** On a successful output operation, your program received a request-to-write indication. The partner program will send data as soon as possible. You should allow the partner program to send this data.

Action: Issue an input operation as soon as possible.

0011 **Description:** Your program received control data, sent with the program start request, on the first input operation, along with a detach indication. The communications transaction with the partner

program has ended, but the session with the partner system is still active.

Action: Your program can either issue an end-of-session function or end.

0013 **Description:** Your program received control data, sent with the program start request, on the first input operation, along with a turnaround indication. In addition, the partner program requested confirmation.

Action: Process any control data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an output operation. If your program does detect an error, issue a fail function, or end your program.

0014 **Description:** On a successful input operation, your program received some data with a turnaround indication. In addition, the partner program requested confirmation.

Action: Process any data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an output operation. If your program does detect an error, issue a fail function, or end your program.

0015 **Description:** On a successful input operation, your program received some data. In addition, the partner program requested confirmation.

Action: Process any data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program.

0018 **Description:** Your program received control data, sent with the program start request, on the first input operation. In addition, the partner program requested confirmation.

Action: Process any data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program.

001C **Description:** On a successful input operation, your program received some data with a detach indication. In addition, the partner program requested confirmation.

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and then:

- If your program started the transaction, it can issue another evoke operation (to start another program), issue a release operation (to perform local processing or to start another session), or end.
- If a program start request from the partner program started the transaction, your program can issue an end-of-session function or end.

If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. To end the transaction abnormally if your program detects an error, issue an end-of-session function, or end your program.

001D **Description:** Your program received control data, sent with the program start request, on the first input operation, along with a detach indication. In addition, the partner program requested confirmation.

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and can then issue an end-of-session function or end.

If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. To end the transaction abnormally if your program detects an error, issue an end-of-session function, or end your program.

0044 **Description:** On a successful input operation, your program received control data with a turnaround indication. In addition, the partner program requested confirmation.

Action: Process any data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an output operation. If your program does detect an error, issue a fail function, or end your program.

0045 **Description:** On a successful input operation, your program received some control data. In addition, the partner program requested confirmation.

Action: Process any control data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program.

0046 **Description:** On a successful input operation, your program received some control data with a detach indication. In addition, the partner program requested confirmation.

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and then:

- If your program started the transaction, it can issue another evoke operation (to start another program), issue a release operation (to perform local processing or to start another session), or end.
- If a program start request from the partner program started the transaction, your program can issue an end-of-session function or end.

If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. To end the

transaction abnormally if your program detects an error, issue an end-of-session function, or end your program.

Major Code 02

Major Code 02 – Input operation completed successfully, but your job is being ended (controlled).

Description: The input operation issued by your program completed successfully. Your program may have received some data or a message from the remote system. However, your job is being ended (controlled).

Action: Your program should complete its processing and end as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

Code	Description/Action
0200	<p>Description: On a successful input operation, your program received some data with a turnaround indication. Also, your job is being ended (controlled). The partner program is ready to receive data from your program.</p> <p>Action: Your program can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.</p>
0201	<p>Description: On a successful input operation, your program received some data. Also, your job is being ended (controlled). Your program can continue to receive data until it receives a turnaround indication (which allows your program to send data) or a detach indication.</p> <p>Action: Your program can issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.</p>
0202	<p>Description: Your program received control data, sent with the program start request, on the first input operation. Also, your job is being ended (controlled). Your program can continue to receive data until it receives a turnaround indication (which allows your program to send data) or a detach indication.</p> <p>Action: Your program can issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.</p>

- 0204** **Description:** On a successful input operation, your program received some control data with a turnaround indication. Also, your job is being ended (controlled). The partner program is ready to receive data from your program.
- Action:** Your program can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
- 0205** **Description:** On a successful input operation, your program received some control data. Also, your job is being ended (controlled). Your program can continue to receive data until it receives a turnaround indication (which allows your program to send data) or a detach indication.
- Action:** Your program can issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
- 0206** **Description:** Your program received control data, sent with the program start request, on the first input operation, along with a turnaround indication. Also, your job is being ended (controlled). The partner program is ready to receive data from your program.
- Action:** Your program can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
- 0208** **Description:** On a successful input operation, your program received a detach indication with the last of the data. The communications transaction with the partner program has ended, but the session with the partner system is still active. Also, your job is being ended (controlled).
- Action:** If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the partner program started the transaction, your program can either issue an end-of-session function or end. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
- 0211** **Description:** Your program received control data, sent with the program start request, on the first input operation, along with a detach indication. The communications transaction with the partner program has ended, but the session with the partner system is still active. Also, your job is being ended (controlled).
- Action:** Your program can either issue an end-of-session function or end. However, the recommended action is to complete all processing and end your program as soon as possible. The system

eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

020C **Description:** On a successful input operation, your program received a detach indication with the last of the control data. The communications transaction with the partner program has ended, but the session with the partner system is still active. Also, your job is being ended (controlled).

Action: If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the partner program started the transaction, your program can either issue an end-of-session function or end. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0213 **Description:** Your program received control data, sent with the program start request, on the first input operation, along with a turnaround indication. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: Process any control data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an output operation. If your program does detect an error, issue a fail function, or end your program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0214 **Description:** On a successful input operation, your program received some data with a turnaround indication. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: Process any data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an output operation. If your program does detect an error, issue a fail function, or end your program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0215 **Description:** On a successful input operation, your program received some data. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: Process any data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0218

Description: Your program received control data, sent with the program start request, on the first input operation. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: Process any data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

021C

Description: On a successful input operation, your program received some data with a detach indication. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and then:

- If your program started the transaction, it can issue another evoke operation (to start another program), issue a release operation (to perform local processing or to start another session), or end.
- If a program start request from the partner program started the transaction, your program can issue an end-of-session function or end.

If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. If your program detects an error and wants to end the transaction abnormally, issue an end-of-session function, or end your program.

However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

021D

Description: Your program received control data, sent with the program start request, on the first input operation, along with a detach indication. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and can then issue an end-of-session function or end. If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. To end a transaction abnormally if your program detects an error, issue an end-of-session function, or end your program.

However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0244 **Description:** On a successful input operation, your program received some control data with a turnaround indication. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: Process any control data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an output operation. If your program does detect an error, issue a fail function, or end your program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0245 **Description:** On a successful input operation, your program received some control data. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: Process any data received with the request. If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0246 **Description:** On a successful input operation, your program received some control data with a detach indication. In addition, the partner program requested confirmation. Also, your job is being ended (controlled).

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and then:

- If your program started the transaction, it can issue another evoke operation (to start another program), issue a release operation (to perform local processing or to start another session), or end.
- If a program start request from the partner program started the transaction, your program can issue an end-of-session function or end.

If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. To end the transaction abnormally if your program detects an error, issue an end-of-session function, or end your program.

However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

Major Code 03

Major Code 03 – Input operation completed successfully, but no data received.

Description: The input operation issued by your program completed successfully, but no data was received.

Action: Examine the minor return code and continue with the next operation.

Code	Description/Action
0300	<p>Description: On a successful input operation, your program received a turnaround indication without any data. The session is still active.</p> <p>Action: Issue an output operation.</p>
0301	<p>Description: On a successful input operation, your program received no data. Your program must continue to receive input until it receives a turnaround or detach indication.</p> <p>Action: Issue an input operation.</p>
0302	<p>Description: Your program received a control data record with a data length of zero, sent with the program start request, on the first input operation. Your program must continue to receive input until it receives a turnaround or detach indication.</p> <p>Action: Issue an input operation.</p>
0304	<p>Description: On a successful input operation, your program received a turnaround indication with a control data record with a data length of zero. The session is still active.</p> <p>Action: Issue an output operation.</p>
0305	<p>Description: On a successful input operation, your program received a control data record with a data length of zero. Your program must continue to receive input until it receives a turnaround or detach indication.</p> <p>Action: Issue an input operation.</p>
0306	<p>Description: Your program received a control data record with a data length of zero, sent with the program start request, on the first input operation, along with a turnaround indication. The session is still active.</p> <p>Action: Issue an output operation.</p>
0308	<p>Description: On a successful input operation, your program received a detach indication without any data. The communications transaction with the partner program has ended, but the session with the partner system is still active.</p> <p>Action: If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the partner program started the trans-</p>

action, your program can either issue an end-of-session function or end.

0309 **Description:** On a read-from-invited-program-devices operation, your program did not receive any data. Also, your job is being ended (controlled).

Action: Your program can continue processing. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

Messages:

CPF4741 (Notify)

0311 **Description:** Your program received a control data record with a data length of zero, sent with the program start request, on the first input operation, along with a detach indication. The communications transaction with the partner program has ended, but the session with the partner system is still active.

Action: Your program can either issue an end-of-session function or end.

030C **Description:** On a successful input operation, your program received a control data record with a data length of zero, along with a detach indication. The communications transaction with the partner program has ended, but the session with the partner system is still active.

Action: If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the partner program started the transaction, your program can either issue an end-of-session function or end.

0310 **Description:** On a read-from-invited-program-devices operation, the time interval specified by a timer function in your program or by the WAITRCD value specified for the ICF file expired.

Action: Issue the intended operation after the specified time interval has ended. For example, if you were using the time interval to control the length of time to wait for data, you can issue another read-from-invited-program-devices operation to receive the data.

Note: Since no specific program device name is associated with the completion of this operation, the program device name in the common I/O feedback area is set to *N. Therefore, your program should not make any checks based on the program device name after receiving the 0310 return code.

Messages:

CPF4742 (Status)

CPF4743 (Status)

0313 **Description:** Your program received a control data record with a data length of zero, sent with the program start request, on the first input operation, along with a turnaround indication. In addition, the partner program requested confirmation.

Action: If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input or output operation. If your program does detect an error, issue a fail function, or end your program.

0314 **Description:** On a successful input operation, your program received a turnaround indication without any data. In addition, the partner program requested confirmation.

Action: If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input or output operation. If your program does detect an error, issue a fail function, or end your program.

0315 **Description:** On a successful input operation, your program did not receive any data. In addition, the partner program requested confirmation.

Action: If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program.

0318 **Description:** Your program received a control data record with a data length of zero, sent with the program start request, on the first input operation. In addition, the partner program requested confirmation.

Action: If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program.

031C **Description:** On a successful input operation, your program received a detach indication without any data. In addition, the partner program requested confirmation.

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and then:

- If your program started the transaction, it can issue another evoke operation (to start another program), issue a release operation (to perform local processing or to start another session), or end.
- If a program start request from the partner program started the transaction, your program can issue an end-of-session function or end.

If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. If your program detects an error and wants to end the transaction abnormally, issue an end-of-session function, or end your program.

031D **Description:** Your program received a control data record with a data length of zero, sent with the program start request, on the first input operation, along with a detach indication. In addition, the partner program requested confirmation.

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and can then issue an end-of-session function or end. If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. To end the transaction abnormally if your program detects an error, issue an end-of-session function, or end your program.

0344 **Description:** On a successful input operation, your program received a control data record with a data length of zero, along with a turn-around indication. In addition, the partner program requested confirmation.

Action: If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input or output operation. If your program does detect an error, issue a fail function, or end your program.

0345 **Description:** On a successful input operation, your program received a control data record with a data length of zero. In addition, the partner program requested confirmation.

Action: If your program detects no errors, respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, then issue an input operation. If your program does detect an error, issue a fail function, or end your program.

0346 **Description:** On a successful input operation, your program received a control data record with a data length of zero, along with a detach indication. In addition, the partner program requested confirmation.

Action: If your program detects no errors, it should respond to the confirm request with a respond-to-confirm (RSPCONFIRM) function, and then:

- If your program started the transaction, it can issue another evoke operation (to start another program), issue a release operation (to perform local processing or to start another session), or end.
- If a program start request from the partner program started the transaction, your program can issue an end-of-session function or end.

If your program does detect an error, issue a fail operation. The transaction remains active, and your program and the partner program can perform the necessary error recovery. To end the transaction abnormally if your program detects an error, issue an end-of-session function, or end your program.

Major Code 04

Major Code 04 – Output exception occurred.

Description: An output exception occurred because your program attempted to send data when it should be receiving data. The data from your output operation was not sent. You can attempt to send the data later.

Action: Issue an input operation to receive the data.

Code	Description/Action
0402	<p>Description: Your program tried to select the USRDFN, SFL, or SFLCTL keyword on a record format, but these keywords are not supported for the program device.</p> <p>Action: Correct the error that caused your program to select a keyword that was not valid.</p> <p>Messages:</p> <p>CPF5064 (Notify)</p>
0412	<p>Description: An output exception occurred because your program attempted to send data when it should have been receiving data that was sent by the partner program. The data from your output operation was not sent to the partner system. Your program can attempt to send the data later.</p> <p>Action: Issue an input operation to receive the data.</p> <p>Messages:</p> <p>CPF4750 (Notify) CPF5076 (Notify)</p>

Major Codes 08 and 11

Major Codes 08 and 11 — Miscellaneous program errors occurred.

Description: The operation just attempted by your program was not successful. The operation may have failed because it was issued at the wrong time.

Action: Refer to the minor code description for the appropriate recovery action.

Code	Description/Action
0800	<p>Description: The acquire operation just attempted by your program was not successful. Your program tried to acquire a program device that was already acquired and is still active.</p> <p>Action: If the session associated with the original acquire operation is the one needed, your program can begin communicating in that session since it is already available. If you want a different session, issue another acquire operation for the new session by specifying a different program device name in the PGMDEV parameter of the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command that precedes the program.</p> <p>Messages:</p> <ul style="list-style-type: none">CPD4077 (Diagnostic)CPF5041 (Status)CPF50A0 (Status)
1100	<p>Description: The read-from-invited-program-devices operation just attempted by your program was not successful because your program tried this operation when no program devices were invited and no timer function was in effect.</p> <p>Action: Issue an invite function (or a combined operation that includes an invite) followed by a read-from-invited-program-devices operation.</p> <p>Messages:</p> <ul style="list-style-type: none">CPF4740 (Notify)

Major Code 34

Major Code 34 – Input exception occurred.

Description: The input operation attempted by your program was not successful. The data received was too long for your program's input buffer or was not compatible with the record format specified on the input operation.

Action: Refer to the minor code description for the appropriate recovery action.

Code	Description/Action
3421	<p>Description: The input operation issued by your program was not successful because the length of the control data received exceeded your program's input record length. The control data received was truncated. The overflow data (excess data received) will be discarded.</p> <p>Action: Your program can issue another input operation to receive the next record or any control information that may have been sent with the truncated data. For example, if the partner program sent a detach indication with the data, the next input operation completes with a return code of 0308. However, the recommended action is to either close the ICF file, end your program, and then change your program so that the input record length is at least as long as the longest data record to be received; or specify OVRFLWDTA(*RETAIN) on the CHGICFDEVE or OVRICFDEVE command to retain overflow data.</p> <p>Messages:</p> <p>CPF4936 (Notify)</p>
3422	<p>Description: Your program received a control data record with the program start request. Your program tried to receive the control data record on the first input operation. The operation was not successful because the length of the control data received exceeded your program's input record length. The control data received was truncated. The overflow data (excess data received) will be discarded.</p> <p>Action: Your program can issue another input operation to receive the next record or any control information that may have been sent with the truncated data. For example, if the partner program sent a detach indication with the data, the next input operation completes with a return code of 0308. However, the recommended action is to close the ICF file, end your program, and then change your program so that the input record length is at least as long as the longest data record to be received; or specify OVRFLWDTA(*RETAIN) on the CHGICFDEVE or OVRICFDEVE command to retain overflow data.</p> <p>Messages:</p> <p>CPF4936 (Notify)</p>

- 3431** **Description:** The input operation issued by your program was not successful because the length of the data received exceeded your program's input record length. The data received was truncated. The overflow data (excess data received) will be discarded.
- Action:** Your program can issue another input operation to receive the next record or any control information that may have been sent with the truncated data. For example, if the partner program sent a detach indication with the data, the next input operation completes with a return code of 0308. However, the recommended action is to either close the ICF file, end your program, and then change your program so that the input record length is at least as long as the longest data record to be received; or specify OVRFLWDTA(*RETAIN) on the CHGICFDEVE or OVRICFDEVE commands to retain overflow data.
- Messages:**
- CPF4911 (Notify)
- 3441** **Description:** A valid record format name was specified with format selection type *RMTFMT or *RECID. However, although the data received matched one of the record formats in the ICF file, it did not match the format specified on the read operation.
- Action:** Correct your program to issue a read operation that does not specify a record format name, or specify the correct record format name to process the data based on the format selection option for the file.
- Messages:**
- CPF5058 (Notify)
- 3451** **Description:** Your program specified a file record size that was not large enough for the indicators to be included with the data sent by the partner program (for a file defined with a nonseparate indicator area). Your program did not receive any data. For a file using a nonseparate indicator area, the actual record length field in the device-dependent I/O feedback area contains the number of indicators specified by the record format.
- Action:** End the session; close the file; correct the file record size; then open the file again.
- Messages:**
- CPF4768 (Notify)
- 3461** **Description:** The input operation issued by your program was not successful because the partner system sent a Fail (FMH7) return code before completing the logical record. This error occurs only on basic conversations.
- Action:** Issue an input operation to receive the Fail (FMH7) return code, and perform any recovery associated with the description.
- Messages:** No exception is created on this return code.
- 3471** **Description:** The input operation issued by your program was not successful because the length of the data received exceeded your program's input record length. The data received was truncated.

The overflow data (excess data received) has been retained and will be returned on subsequent input operations.

Action: Issue a read operation to receive the overflow data along with any control information that may have been sent.

Messages:

CPF4935 (Notify)

3481

Description: The input operation issued by your program was not successful because the length of the control data received exceeded your program's input record length. The control data received was truncated. The overflow control data (excess control data received) has been retained and will be returned on subsequent input operations.

Action: Issue a read operation to receive the overflow control data along with any control information that may have been sent.

Messages:

CPF4937 (Notify)

Major Code 80

Major Code 80 — Permanent system or file error (irrecoverable).

Description: An irrecoverable file or system error has occurred. The underlying communications support may have ended and your session has ended. If the underlying communications support ended, it must be established again before communications can resume. Recovery from this error is unlikely until the problem causing the error is detected and corrected.

Action: You can perform the following general actions for all 80xx return codes. Specific actions are given in each minor code description.

- Close the file, open the file again, then establish the session. If the operation is still not successful, your program should end the session.
- Continue local processing.
- End.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code	Description/Action
8081	<p>Description: The operation attempted by your program was not successful because a system error condition was detected.</p> <p>Action: Your communications configurations may need to be varied off and then on again. Your program can do one of the following:</p> <ul style="list-style-type: none">• Continue local processing.• Close the ICF file, open the file again, and establish the session again.• End. <p>Messages:</p> <ul style="list-style-type: none">CPF4061 (Diagnostic)CPF4170 (Escape)CPF4323 (Escape)CPF4510 (Escape)CPF5257 (Escape)CPF5411 (Escape)CPF5424 (Escape)CPF5425 (Escape)CPF5449 (Escape)CPF5537 (Escape)
8082	<p>Description: The operation attempted by your program was not successful because the device supporting communications between your program and the partner location is not usable. For example, this may have occurred because communications were stopped for the device by a Hold Communications Device (HLDCMNDEV) command. Your program should not issue any operations to the device.</p> <p>Action: Communications with the remote program cannot resume until the device has been reset to a varied on state. If the device has been held, use the Release Communications Device (RLSCMNDEV)</p>

command to reset the device. If the device is in an error state, vary the device off and then on again. Your program can attempt to establish the session again, continue local processing, or end.

Messages:

CPF4744 (Escape)
CPF5269 (Escape)

80B3

Description: The open operation issued by your program was not successful because the ICF file is in use by another process.

Action: Wait for the file to become available, then issue another open operation. Otherwise, your program may continue processing, or it can end.

Consider increasing the WAITFILE parameter with the Change ICF File (CHGICFF) or Override ICF File (OVRICFF) command to allow more time for the file resources to become available.

Messages:

CPF4128 (Escape)

80C0

Description: An irrecoverable error has occurred on the session. The session was ended abnormally either by the partner system or because of a partner protocol error.

Note: On a mapped conversation, return codes 81C5 and 81C6 are translated into 80C0.

Action: Contact the partner system to determine why the session was ended. Your program can continue local processing or end.

Messages:

CPF4145 (Escape)
CPF5107 (Escape)
CPF5280 (Escape)
CPF5282 (Escape)
CPF5283 (Escape)
CPF5393 (Escape)
CPF5394 (Escape)
CPF5424 (Escape)
CPF5527 (Escape)
CPF5529 (Escape)

80D0

Description: The evoke function issued by your program was not successful because the program specified on the evoke function is not available. The session has not ended, but the evoke function cannot be issued again.

Action: Contact the partner system to determine why the program specified on the evoke function is not available.

Messages:

CPF5395 (Escape)
CPF5531 (Escape)

80EB

Description: The open operation attempted by your program was not successful due to one of the following:

- Your program used an option of update or delete to open the file, but that option is not supported by the program device.

- Your program requested both blocked data and user buffers on an open option, but these formats cannot be selected together.
- Your program tried to open a source file, but the file was not created as a source file.
- There is a mismatch on the INDARA keyword between your program and the ICF file as to whether or not a separate indicator area should be used.
- The file was originally opened as a shared file; however, no program devices were ever acquired for the file before your program attempted the current open operation.

Action: After performing one of the following actions, your program can try the open operation again:

- If the update and delete options are not supported for the program device, use an option of input, output, or both.
- If your program tried selecting user buffers and blocked data together, it should try selecting one or the other, but not both.
- If your program tried to open a non-source file as a source file, either change the file name or change the library name.
- If there was a mismatch on the INDARA keyword, either correct the file or correct your program so that the two match.
- If no program devices were previously acquired for a shared file, acquire one or more program devices for the file.

Messages:

CPF4133 (Escape)
 CPF4156 (Escape)
 CPF4238 (Escape)
 CPF4250 (Escape)
 CPF4345 (Escape)
 CPF5522 (Escape)
 CPF5549 (Escape)

80ED

Description: The open operation attempted by your program was not successful due to one of the following:

- There is a record format level mismatch between your program and the ICF file.
- Your program tried to use a temporary file close option, but this option is not allowed for this file type.

Action:

- If there was a file-level mismatch, first close the file, then compile your program again to match the file level of the ICF file, or change or override the file to LVLCHK(*NO). Then open the file again.
- If your program tried to use a temporary file close option that was not valid, try a permanent close instead. If the problem continues, begin problem analysis using the ANZPRB command.

Messages:

CPF4131 (Escape)

80EF

Description: Your program attempted an open operation on a file or library for which the user is not authorized.

Action: Close the file. Either change the file or library name on the

open operation, or obtain authority for the file or library from your security officer. Then issue the open operation again.

Messages:

CPF4104 (Escape)

80F8

Description: The open operation attempted by your program was not successful because one of the following occurred:

- The file is already open.
- The file is marked in error on a previous return code.

Action:

- If the file is already open, close the file and end your program. Remove the duplicate open operation from your program, then issue the open operation again.
- If the file is marked in error, your program can check the job log to see what errors occurred previously, then take the appropriate recovery action for those errors.

Messages:

CPF4132 (Escape)

CPF5129 (Escape)

Major Code 81

Major Code 81 – Permanent session error (unrecoverable).

Description: An unrecoverable session error occurred during an I/O operation. Your session cannot continue and has ended. Before communications can resume, the session must be established again by using an acquire operation or another program start request. Recovery from this error is unlikely until the problem causing the error is detected and corrected. Operations directed to other sessions associated with the file should work.

Action: You can perform the following general actions for all 81xx return codes. Specific actions are given in each minor return code description.

If your program initiated the session, you can:

- Correct the problem and establish the session again. If the operation is still not successful, your program should end the session.
- Continue processing without the session.
- End.

If your session was initiated by a program start request from the partner program, you can:

- Continue processing without the session.
- End.

Several of the minor codes indicate that an error condition must be corrected by changing a value in the communications configuration or in the file.

- To change a parameter value in the communications configuration, vary the configuration off, make the change to the configuration description, then vary the configuration on.
- To change a parameter value in the file, use the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

Note: When a parameter can be specified both in the ADDICFDEVE or OVRICFDEVE command and in the configuration, the value in the ADDICFDEVE or OVRICFDEVE command overrides the value specified in the configuration (for your program only). Therefore, in some cases, you may choose to make a change with the ADDICFDEVE or OVRICFDEVE command rather than in the configuration.

Several other minor codes indicate a line or remote system error and may require an operator to correct the error.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code	Description/Action
8140	<p>Description: A cancel reply was received from your program or from the operator in response to a notify message, or was the result of a system default, causing the session to be ended. The session is no longer active.</p> <p>Action: If your program started the session, issue an acquire operation to start the session again. If your program was started by a program start request, it can continue local processing or end.</p> <p>Messages:</p> <p style="padding-left: 40px;">CPF5104 (Escape)</p>
8191	<p>Description: A network interface, permanent line, or controller error occurred on an input or output operation, and the system operator attempted recovery in response to the error message. You can learn what type of error occurred by checking the system operator's message queue. The session has ended. Data may have been lost.</p> <p>Action: If your program started the session, issue an acquire operation to start the session again. If your program was started by a program start request from the partner program, it can continue local processing or end.</p> <p>Messages:</p> <p style="padding-left: 40px;">CPF4146 (Escape) CPF410E (Escape) CPF4155 (Escape) CPF4193 (Escape) CPF4291 (Escape) CPF510E (Escape) CPF5128 (Escape) CPF5198 (Escape) CPF5544 (Escape)</p>
8196	<p>Description: The partner system sent a Systems Network Architecture (SNA) UNBIND command to your system, or the session was ended locally.</p> <p>Action: To start another session, issue the acquire operation again. Otherwise, your program can continue local processing or end.</p> <p>Messages:</p> <p style="padding-left: 40px;">CPF4060 (Diagnostic) CPF4145 (Escape) CPF4299 (Escape) CPF5107 (Escape) CPF5166 (Escape) CPF5524 (Escape)</p>
8197	<p>Description: On an input or output operation, the partner system ended the transmission abnormally because it could not continue the session. The session has ended.</p> <p>Action: If your program started the session, issue an acquire operation to start the session again. If your program was started by a program start request from the partner program, it can continue local processing or end.</p>

Messages:

CPF4145 (Escape)
CPF5107 (Escape)
CPF5283 (Escape)
CPF5526 (Escape)

81C2 **Description:** The operation issued by your program was not successful because the session is not active.

Action: Issue another acquire operation, continue local processing, or end your program.

Messages:

CPF5396 (Escape)

81C5 **Description:** The partner program or partner system abnormally ended the session (TYPE = SVC).

Action: Contact the remote system to determine why the error occurred.

Messages:

CPF4145 (Escape)
CPF5107 (Escape)
CPF5283 (Escape)

81C6 **Description:** The partner program or partner system abnormally ended the session (TYPE = TIMER). For example, the partner program may have been canceled by the operator.

Action: Contact the remote system to determine why the error occurred.

Messages:

CPF4145 (Escape)
CPF5107 (Escape)
CPF5283 (Escape)

81E9 **Description:** An input operation was issued and the format selection option for the ICF file was *RECID, but the data received did not match any record formats in the file. There was no format in the file defined without a RECID keyword, so there was no default record format to use. The session has ended.

Action: Verify that the data sent by the partner program was correct. If the data was not correct, have the operator on the partner system change the partner program to send the correct data. If the data was correct, add a RECID keyword definition to the file that matches the data, or define a record format in the file without a RECID keyword so that a default record format can be used on input operations. If your program started the session, use another acquire operation to start the session again. If a program start request started your program, continue local processing or end.

Messages:

CPF5291 (Escape)

Major Code 82

Major Code 82 – Open or acquire operation failed.

Description: Your attempt to establish a session was not successful. The error may be recoverable or permanent, and recovery from it is unlikely until the problem causing the error is detected and corrected.

Action: You can perform the following general actions for all 82xx return codes. Specific actions are given in each minor code description.

If your program was attempting to start the session, you can:

- Correct the problem and attempt to establish the session again. The next operation could be successful only if the error occurred because of some temporary condition such as the communications line being in use at the time. If the operation is still not successful, your program should end.
- Continue processing without the session.
- End.

If your session was initiated by a program start request from the partner program, you can:

- Correct the problem and attempt to connect to the requesting program device again. If the operation is still not successful, your program should end.
- Continue processing without the session.
- End.

Several of the minor codes indicate that an error condition must be corrected by changing a value in the communications configuration or in the file.

- To change a parameter value in the communications configuration, vary the configuration off, make the change to the configuration description, then vary the configuration on.
- To change a parameter value in the file, use the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

Note: When a parameter can be specified both in the ADDICFDEVE or OVRICFDEVE command and in the configuration, the value in the ADDICFDEVE or OVRICFDEVE command overrides the value specified in the configuration (for your program only). Therefore, in some cases, you may choose to make a change with the ADDICFDEVE or OVRICFDEVE command rather than in the configuration.

If no changes are needed in your file or in the configuration (and depending on what the return code description says):

- If the attempted operation was an acquire, issue the acquire operation again.
- If the attempted operation was an open, close the file and issue the open operation again.

Code	Description/Action
8209	<p>Description: The open or acquire operation issued by your program was not successful because a prestart job is being canceled. One of the following may have occurred:</p> <ul style="list-style-type: none"> • An End Job (ENDJOB), End Prestart Job (ENDPJ), End Subsystem (ENDSBS), End System (ENDSYS), or Power Down System (PWRDWNSYS) command was being issued. • The maximum number of prestart jobs (MAXJOBS parameter) was reduced by the Change Prestart Job Entry (CHGPJE) command. • The value for the maximum number of program start requests allowed (specified in the MAXUSE parameter on the ADDPJE or CHGPJE command) was exceeded. • Too many unused prestart jobs exist. • The prestart job had an initialization error. <p>Action: Complete all processing and end your program as soon as possible. Correct the system error before starting this job again.</p> <p>Messages:</p> <p style="padding-left: 40px;">CPF4292 (Escape) CPF5313 (Escape)</p>
8233	<p>Description: A program device name that was not valid was detected. Either an ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command was not run, or the program device name in your program does not match the program device name specified in the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command for the session being acquired. The session was not started.</p> <p>Action: If the error was in your program, change your program to specify the correct program device name. If an incorrect identifier was specified in the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, specify the correct value in the PGMDEV parameter.</p> <p>Messages:</p> <p style="padding-left: 40px;">CPF4288 (Escape) CPF5068 (Escape)</p>
8281	<p>Description: On an unsuccessful open or acquire operation, a system error condition was detected. For example, the file may previously have been in error, or the file could not be opened due to a system error.</p> <p>Action: Your communications configurations may need to be varied off and then on again. Your program can do one of the following:</p> <ul style="list-style-type: none"> • Continue local processing. • Close the ICF file, open the file again, and acquire the program device again. However, if this results in another 8281 return code, your program should close the file and end. • Close the file and end. <p>Messages:</p> <p style="padding-left: 40px;">CPF4106 (Escape) CPF4168 (Escape) CPF4182 (Escape) CPF4369 (Escape)</p>

CPF4370 (Escape)
CPF4375 (Escape)
CPF5257 (Escape)
CPF5274 (Escape)
CPF5317 (Escape)
CPF5318 (Escape)
CPF5507 (Escape)
CPF5536 (Escape)

8282

Description: The open or acquire operation attempted by your program was not successful because the device supporting communications between your program and the partner location is not usable. For example, this may have occurred because communications were stopped for the device by a Hold Communications Device (HLDCMNDEV) command. Your program should not issue any operations to the device. The session was not started.

Action: Communications with the remote program cannot resume until the device has been reset to a varied on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off, then on again. Your program can attempt to acquire the program device again, continue local processing, or end.

Messages:

CPF4298 (Escape)
CPF4354 (Escape)
CPF5269 (Escape)
CPF5358 (Escape)

82A6

Description: On the open or acquire operation attempted by your program, a negative-response with sense data was received when the Systems Network Architecture (SNA) BIND command was sent to the user to start the session. The session was not started.

Action: Close the file. Check for an error in the format of the incorrect BIND command, or contact the partner system to determine why the command failed. After correcting the error, your program can issue the acquire operation again to start the session.

Messages:

CPF4333 (Escape)
CPF5281 (Escape)
CPF5538 (Escape)

82A8

Description: The acquire operation attempted by your program was not successful because the maximum number of program devices allowed for the ICF file has been reached. The session was not started.

Action: Your program can recover by releasing a different program device and issuing the acquire operation again. If more program devices are needed, close the file and increase the MAXPGMDEV value for the ICF file.

Messages:

CPF4745 (Diagnostic)
CPF5041 (Status)

82A9

Description: The acquire operation issued by your program to a *REQUESTER device was not successful due to one of the following causes:

- Your program has already acquired the *REQUESTER device.
- The job was started by a program start request with the *REQUESTER device detached.
- The *REQUESTER device was released because an end-of-session was requested.
- The job does not have a *REQUESTER device; that is, the job was not started by a program start request.
- A permanent error occurred on the session.

Action:

- If the *REQUESTER device is already acquired and your program expects to communicate with the *REQUESTER device, use the program device that acquired the *REQUESTER.
- If the *REQUESTER device is not available and your program expects to communicate with the *REQUESTER device, the partner program must send a program start request without a detach function.
- If your program released its *REQUESTER device, before trying to acquire it, correct the error.
- If this job does not have a *REQUESTER device, correct the error that caused your program to attempt to acquire a *REQUESTER device.
- If a permanent error caused the acquire operation to fail, verify that your program correctly handles the permanent error return codes (80xx, 81xx) it received on previously issued input and output operations. Because your program was started by a program start request, your program cannot attempt error recovery after receiving a permanent error return code. It is the responsibility of the partner program to initiate error recovery.

Messages:

CPF4366 (Escape)
 CPF5380 (Escape)
 CPF5381 (Escape)

82AA

Description: The open or acquire operation attempted by your program was not successful because the remote location name does not match any remote location configured on the system or in the network that matches the remote location name specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. The session was not started.

Action: Your program can continue local processing, or close the file and end. Verify that the name of the remote location is specified correctly in the RMTLOCNAME parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

Messages:

CPF4103 (Escape)
 CPF4363 (Escape)
 CPF4364 (Escape)
 CPF4747 (Escape)
 CPF5378 (Escape)

CPF5379 (Escape)

82AB

Description: The open or acquire operation attempted by your program was not successful because the device description for the remote location was not varied on. The session was not started.

Action: Your program can wait until the communications configuration is varied on and then issue the acquire operation again, it can try the acquire operation again using a different device description, continue local processing, or end.

Messages:

CPF4128 (Escape)

CPF5355 (Escape)

82B3

Description: The open or acquire operation attempted by your program was not successful because none of the sessions specified in the communications configuration is currently available. The session was not started.

Action: Wait for one of the sessions to become available, then issue the acquire operation again. You can use the Work with Configuration Status (WRKCFGSTS) command to determine which job is using each session. Otherwise, your program can continue local processing or end.

Messages:

CPF4128 (Escape)

CPF5355 (Escape)

82B6

Description: You tried to open or acquire a conversation previously acquired in the System/38 system environment.

Action Only ICF files may be used for the *REQUESTER device. Correct your program and try the request again.

Messages:

CPF411A (Escape)

82C3

Description: The open or acquire operation just performed was not successful because one of the following occurred:

- The mode specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command is not defined for the remote location.
- The class of service attached to the specified mode cannot be found.

The session was not started.

Action: The mode and its attached class of service must be configured on the system.

- Verify that the mode was specified correctly in the MODE parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. If the correct name was specified, vary off the specified remote location, define the mode for the specified remote location, vary on the specified remote location, and issue the acquire operation again. Otherwise, specify the correct MODE parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, and issue the acquire operation again.

- Verify that the attached class of service is defined on the system. If it is not defined, either specify a class of service that is already defined, or define the new class of service. Vary the device associated with the remote location off, then on again, and try the request again.

Messages:

CPF4202 (Escape)
 CPF4376 (Escape)
 CPF5383 (Escape)
 CPF5546 (Escape)

82EA

Description: The open or acquire operation attempted by your program was not successful. A format selection of *RECID was specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, but cannot be used with the ICF file because the RECID DDS keyword is not used on any of the record formats in the file. The session was not started.

Action: Close the ICF file. Change the record format selection (FMTSLT) parameter to select formats by some means other than *RECID, or use a file that has a RECID DDS keyword specified for at least one record format. Open the file again.

Messages:

CPF4348 (Escape)
 CPF5521 (Escape)

82EC

Description: The acquire operation attempted by your program was not successful because CNVTYPE(*USER) does not support FMTSLT(*RMTFMT).

Action: End your program, correct the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, then run your program again.

Messages:

CPF4349 (Escape)
 CPF5541 (Escape)

82EE

Description: Your program attempted an open or acquire operation to a device that is not supported. Your program tried to acquire a device that is not a valid ICF communications type, or it is trying to acquire the requesting program device in a program that was not started by a program start request. The session was not started.

Action: Your program can continue local processing or end. Verify that the name of the remote location is specified correctly in the RMTLOCNAME parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. If your program was attempting to acquire a non-ICF device, use the appropriate interface for that communications type. If your program was attempting to acquire a requesting program device, verify that your program is running in the correct environment.

Messages:

CPF4105 (Escape)
 CPF4223 (Escape)
 CPF4251 (Escape)
 CPF4760 (Escape)

CPF5038 (Escape)
CPF5550 (Escape)

82EF

Description: Your program attempted an acquire operation (or an open operation that implicitly acquires a session) to a device that the user is not authorized to, or that is in service mode. The session was not started.

Action: If the operation was an acquire, correct the problem and issue the acquire again. If the operation was an open, close the file, correct the problem, then issue the open operation again. To correct an authority error, obtain authority for the device from your security officer or device owner. If the device is in service mode, wait until machine service function (MSF) is no longer using the device before issuing the operation again.

Messages:

CPF4104 (Escape)
CPF4186 (Escape)
CPF5278 (Escape)
CPF5279 (Escape)

82F0

Description: The open or acquire operation attempted by your program to a requesting program device was not successful because there is an error in the ICF file.

Action: End your program, correct the error, then have the partner program send the program start request again.

Messages:

CPF4324 (Escape)
CPF5540 (Escape)

82F2

Description: The acquire operation issued by your program was not successful because the CNVTYPE specification on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command was not valid.

Action: End your program, correct the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, then run your program again.

Messages:

CPF4350 (Escape)
CPF4351 (Escape)
CPF5542 (Escape)
CPF5543 (Escape)

82F4

Description: The open or acquire operation attempted by your program was not successful because the open operation for *input only* is valid only for a requesting program device.

Action: End your program, correct the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, then run your program again.

Messages:

CPF4322 (Escape)
CPF5539 (Escape)

Major Code 83

Major Code 83 – Session error occurred (the error is recoverable).

Description: A session error occurred, but the session may still be active. Recovery within your program might be possible.

Action: You can perform the following general actions for all 83xx return codes. Specific actions are given in each minor code description.

- Correct the problem and continue processing with the session. If the error occurred because of a resource failure on the partner system or because the partner system was not active at the time, a second attempt may be successful. If the operation is still not successful, your program should end the session.
- Issue an end-of-session function and continue processing without the session.
- End.

Several of the minor codes indicate that an error condition must be corrected by changing a value in the communications configuration or in the file.

- To change a parameter value in the communications configuration, vary the configuration off, make the change to the configuration description, then vary the configuration on.
- To change a parameter value in the file, use the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

Note: When a parameter can be specified both in the ADDICFDEVE or OVRICFDEVE command and in the configuration, the value in the ADDICFDEVE or OVRICFDEVE command overrides the value specified in the configuration (for your program only). Therefore, in some cases, you may choose to make a change with the ADDICFDEVE or OVRICFDEVE command rather than in the configuration.

If no changes are needed in your file or in the configuration, and depending on what the return code description says, you should notify the remote location that a change is required at that location to correct the error received.

Code	Description/Action
830B	<p>Description: Your program attempted an operation that was not valid because the session was not yet acquired or has ended. The session may have ended because of a release operation, an end-of-session function, or a permanent error. Your program may have incorrectly handled a previous error.</p> <p>Action: Verify that your program does not attempt any operations without an active session. Also verify that your program correctly handles the permanent error or session-not-acquired return codes (80xx, 81xx, 82xx) it received on previously issued input and output operations. To recover from an incorrectly handled error condition, your program may or may not be able to issue another acquire operation, depending on the return code.</p>

Messages:

CPD4079 (Diagnostic)
CPF4739 (Status)
CPF5067 (Escape)
CPF5068 (Escape)
CPF5070 (Escape)

8316

Description: The evoke function issued by your program was not successful because the target program could not be found.

Action: Verify that your program specified the correct target program name on the evoke function. If necessary, contact the partner location to determine if the program still exists on the remote system.

Messages:

CPF4734 (Notify)

831E

Description: The operation attempted by your program was not valid, or a combination of operations that was not valid was specified. The session is still active. The error may have been caused by one of the following:

- Your program issued an operation that is not recognizable or not supported by APPC.
- Your program requested a combination of operations or keywords that was not valid, such as a combined write-then-read operation with the invite function specified.
- Your program issued an input operation, or an output operation with the invite or allow-write function, for a file that was opened for output only.
- Your program issued an output operation for a file that was opened for input only.
- Your program issued a close operation with a temporary close option.

Action: Your program can try a different operation, issue a release operation or end-of-session function, or end. Correct the error in your program before trying to communicate with the partner program.

If the file was opened for input only, do not issue any output operations; or, if the file was opened for output only, do not issue any input operations, and do not use the invite or allow-write function on an output operation. If such an operation is needed, then release the session, close the ICF file, and open the file again for input and output.

Messages:

CPF4764 (Notify)
CPF4766 (Notify)
CPF4790 (Notify)
CPF5132 (Escape)
CPF5149 (Escape)

831F

Description: Your program specified data or a length for the operation that was not valid; however, the session is still active. One of the following caused the error indication:

- On an output operation, your program tried to send a data record that was longer than the MAXRCDLEN value specified for the ICF file.
- The program used a read or write operation that specified a data length greater than the record format in the ICF file.
- If this was a timer function, the format of the timer interval was not HHMMSS.
- If a system-defined format was used to specify the operation, or if the variable-length-data-record (VARLEN) function was used, then the length of the user buffer was not valid.

Action: If you want your program to recover, try the operation again with a smaller data length. If you do not need your program to recover immediately, do one of the following:

- Change the record format length in the ICF file, or change the record length in your program and compile your program again.
- For an input operation, specify a data length equal to or less than the record format length, or do not specify a length at all.
- If the timer function was used, verify that the format of the timer interval is HHMMSS.
- For an output operation that used the variable-length-data-record (VARLEN) function, verify that the length specified is less than the record length specified for the ICF file when it was opened.

Messages:

CPF4762 (Notify)
 CPF4765 (Notify)
 CPF4767 (Notify)

8327

Description: The input or output operation issued by your program was not successful because there was no active transaction. Either the transaction has ended, or the transaction was never started.

Action: If your program is to start a transaction, it can issue an evoke function. Otherwise, it can issue an end-of-session function or end. If a coding error in your program caused the error, correct your program.

Messages:

CPF5001 (Status)
 CPF5078 (Notify)
 CPF5098 (Notify)
 CPF5277 (Escape)
 CPF5525 (Escape)

8329

Description: An evoke function that was not valid was detected in this session. Your program was started by a program start request and, therefore, cannot issue any evoke functions in this session.

Action: To recover, your program can try a different operation or function. To issue an evoke function in a different session, first issue an acquire operation (using a different program device name), then try the evoke function. Otherwise, your program can issue an end-of-session function, continue local processing, or end. If a coding error caused your program to attempt an evoke function that was not valid, correct your program.

Messages:

CPF5099 (Notify)

832C

Description: A release operation following an invite function was detected. Because your program issued the invite function, it cannot issue a release operation to end the invited session.

Action: Issue an input operation to satisfy the invite function. Otherwise, issue an end-of-session function to end the session. If a coding error caused your program to attempt a release operation that was not valid, correct your program.

Messages:

CPF4769 (Notify)

832D

Description: Following an invite function, your program issued a request-to-write indication or an additional invite function. This operation failed because the original invite function must first be satisfied by an input operation.

Action: Issue an input operation to receive the data that was invited. Otherwise, issue an end-of-session function to end the session. If a coding error caused your program to attempt a request-to-write indication or an additional invite function, correct your program.

Messages:

CPF4924 (Notify)

832F

Description: The evoke function issued by your program was not successful because your program attempted the operation while the current transaction was still active. The operation was not performed, but the session is still active.

Action: Use the detach function to end the current transaction before issuing an evoke function. Correct the error that caused your program to issue an evoke function during an active transaction; then run your program again.

Messages:

CPF5099 (Notify)

8334

Description: The evoke function attempted by your program was not valid. Your program specified a conversation type of *SYS on the CNVTYPE parameter, but it also specified a program name that started with a character whose value was less than X'41'. For CNVTYPE(*SYS), the first character of the program name must be greater than X'40'.

Action: Correct your program so that it issues the evoke correctly, then try the operation again.

Messages:

CPF5015 (Notify)

83C7

Description: On a successful input operation, your program received a fail indication (TYPE=PROG) with no data. No data has been truncated. The partner program may have sent the fail indication to indi-

cate that the previous data it sent or received was in error. The session is still active.

Action: Issue another input operation.

Messages:

CPF5093 (Notify)

83C8 **Description:** On a successful input operation, your program received a fail indication (TYPE=SVC) with no data. No data has been truncated. The partner program may have sent the fail indication to indicate that the previous data it sent or received was in error. The session is still active.

Action: Issue another input operation.

Messages:

CPF4793 (Notify)

83C9 **Description:** On an input or output operation, your program received a fail indication (TYPE=PROG) with or without a confirm indication. Data may have been lost. Data was lost if the fail indication was sent by the partner program before receiving all the data sent by your program. Data was not lost if the fail indication was received on an operation that specified a confirm function, or if the fail indication was sent by the partner program after receiving all the data sent by your program. The partner program may have sent the fail indication to indicate that the data it received was in error. The session is still active, and your program is in receive state.

Action: Issue an input operation.

Messages:

CPF5093 (Notify)

83CA **Description:** On an input or output operation, your program received a fail indication (TYPE=SVC) with or without a confirm indication. Data may have been lost. Data was lost if the fail indication was sent by the partner program before receiving all the data sent by your program. Data was not lost if the fail indication was received on an operation that specified a confirm function, or if the fail indication was sent by the partner program after receiving all the data sent by your program. The partner program may have sent the fail indication to indicate that the data it received was in error. The session is still active, and your program is in receive state.

Action: Issue an input operation.

Messages:

CPF4793 (Notify)

83CB **Description:** On an input operation, your program received a fail indication (TYPE=PROG). The last logical record has been truncated. Truncation occurs when the partner program begins to send a logical record and then sends a fail indication before sending the complete logical record. The partner program may have sent the fail indication to indicate that the previous data it sent is in error. The session is still active.

Action: Issue another input operation.

Messages:

CPF5094 (Notify)

83CC **Description:** On an input operation, your program received a fail indication (TYPE = SVC). The last logical record has been truncated. Truncation occurs when the partner program begins to send a logical record and then sends a fail indication before sending the complete logical record. The partner program may have sent the fail indication to indicate that the previous data it sent is in error. The session is still active.

Action: Issue another input operation.

Messages:

CPF4794 (Notify)

83CD **Description:** The input or output operation issued by your program was not successful because your program attempted a confirm function while it was still in receive state, or because a confirm function was specified for a transaction that was started with a synchronization level of *NONE.

Action: If your program was still in receive state, correct the error that caused your program to attempt the confirm function. If your program is to issue the confirm function while in send state, issue an end-of-session function and change your program to start the transaction with a synchronization level of *CONFIRM.

Messages:

CPF5016 (Notify)

83CE **Description:** The partner location rejected the evoke function issued by your program because the access security information specified on the evoke was not valid. This return code is issued on the current evoke function when the program specifies a confirm function in combination with the evoke function; otherwise, it is returned on a subsequent operation.

Action: Verify that your program specified the correct security information on the evoke function. If necessary, contact the partner location to determine if your access security information is still valid.

Messages:

CPF4734 (Notify)

83CF **Description:** The partner location rejected the evoke function issued by your program because either your program or the partner program does not support the specified conversation type. This return code is issued on the current evoke function when the program specifies a confirm function in combination with the evoke function; otherwise, it is returned on a subsequent operation.

Action: Verify that the correct conversation type was specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command for the program device. If necessary, contact the partner location to determine if the specified conversation type is supported.

Messages:

CPF4734 (Notify)

83D0 **Description:** The partner location rejected the evoke function issued by your program because the program name specified on the evoke is one that the partner location recognizes but currently cannot start. The partner location may have rejected the evoke function because no sessions are currently available. This return code is issued on the current evoke function when the program specifies a confirm function in combination with the evoke function; otherwise, it is returned on a subsequent operation.

Action: If the condition causing the target program to be unavailable is temporary, your program can try the evoke function again.

Messages:

CPF4736 (Notify)

83D1 **Description:** The partner location rejected the evoke function issued by your program because your program specified program initialization parameters (PIP) on the evoke function but the partner program does not allow PIP data. This return code is issued on the current evoke function when the program specifies a confirm function in combination with the evoke function; otherwise, it is returned on a subsequent operation.

Action: Verify that your program specifies PIP data on the evoke function. If it does, contact the partner location to determine if the specified partner program is in error.

Messages:

CPF4734 (Notify)

83D2 **Description:** The partner location rejected the evoke function issued by your program because the partner program has one or more subfields defined in the program initialization parameters (PIP), but your program specified no PIP data on the evoke function, or the evoke function issued by your program specified PIP data that does not correspond in number to those defined for the partner program. This return code is issued on the current evoke function when the program specifies a confirm function in combination with the evoke function; otherwise, it is returned on a subsequent operation.

Action: Verify that your program correctly specifies the PIP data on the evoke function. If necessary, contact the partner location to determine if the PIP data is defined correctly for the specified partner program.

Messages:

CPF4734 (Notify)

83D3 **Description:** The partner location rejected the evoke function issued by your program because the evoke specified a synchronization level that the partner program does not support. This return code is issued on the current evoke function when the program specifies a confirm function in combination with the evoke function; otherwise, it is returned on a subsequent operation.

Action: Verify that your program specifies the correct synchronization level on the evoke function. If necessary, contact the partner location to determine if the synchronization level is defined correctly for the partner program.

Messages:

CPF4734 (Notify)

83D5 **Description:** The partner program has sent a confirm request to your program. Your program must either accept or reject the confirmation request.

Action: The condition may be corrected by one of the following recovery actions:

- If your program detects no errors in the data, respond to the confirm request with a RSPCONFIRM function and then continue.
- If your program does detect an error, reject the confirm request by issuing a fail function, or end the transaction abnormally by issuing an end-of-session function or closing the file. If your program issues a fail function, your program is responsible for the necessary error recovery.
- If your program does not expect to use confirmation processing, have the partner program omit the confirm request.

Messages:

CPF4791 (Notify)

83D6 **Description:** The RSPCONFIRM function issued by your program was not valid because the partner program did not request confirmation, or because the current transaction was started with a synchronization level of *NONE.

Action: If the partner program did not request confirmation, correct the error that caused your program to issue the RSPCONFIRM function. However, if both programs expect to use confirmation processing, the transaction must be started with a synchronization level of *CONFIRM.

Messages:

CPF4792 (Notify)

83E0 **Description:** Your program attempted an operation using a record format that was not defined for the ICF file.

Action: Verify that the name of the record format in your program is correct, then check to see whether the record format is defined in the file definition.

Messages:

CPF5054 (Notify)

83F1 **Description:** Your program closed the ICF file while the transaction was still active. The system abnormally ended the transaction with the partner program.

Action: Examine your program to be sure that it was correct to close the file while the transaction was still active. Normally, a detach indication should be sent or received before the file is closed.

Messages:

CPF4060 (Diagnostic)

- 83F3** **Description:** Your program issued an output operation with an invalid length specification on a basic conversation.
- Action:** Change your program or the record format to issue a correct length on the output operation.
- Messages:**
- CPF5095 (Notify)
 CPF5096 (Notify)
- 83F8** **Description:** Your program attempted to issue an operation to a program device that is marked in error due to a previous I/O or acquire operation. Your program may have handled the error incorrectly.
- Action:** Release the program device, correct the previous error, then acquire the program device again.
- Messages:**
- CPF5293 (Escape)
- 83F9** **Description:** An input operation or an output operation with an allow-write, confirm, invite, or detach function has been requested, but the length specified in the last record indicated that concatenated data is to follow. The data concatenation must be completed before the requested function is allowed.
- Action:** Correct your program to issue an output operation that completes the concatenated data, then start the session again from the beginning.
- Messages:**
- CPF4746 (Notify)

CPI Communications Return Codes

Refer to the *CPI Communications Reference* for information about CPI Communications return codes.

Appendix C. Implementation of the LU Type 6.2 Architecture

This appendix contains a description of the AS/400 implementation of the SNA logical unit type 6.2 architecture. It maps the LU type 6.2 architected verbs and supported parameters to the AS/400 implementation. Each section contains the architected verb and parameters followed by the corresponding ICF functions and commands necessary to support the architected verb. Also included in each section are the DDS and system-supplied format specifications. LU type 6.2 architected verbs that are not supported are not listed.

For information on the CPI Communications implementation of the LU type 6.2 architecture, refer to the *CPI Communications Reference* manual.

AS/400 System Implementation of Control Operator Verbs

This section contains the following types of control operator verbs supported by the AS/400 system:

- Change number of sessions
- Session control
- LU definition

Change-Number-of-Sessions Verbs

The following verbs are in a subcategory of control operator verbs called the change-number-of-sessions, or CNOS, verbs. The CNOS verbs change the session limit, which controls the number of LU-LU sessions per mode name that are available between two LUs for allocation to conversations.

Note: The AS/400 system supports multiple local LUs; therefore, these verbs relate to a particular local LU and a remote LU.

CHANGE_SESSION_LIMIT: The CHANGE_SESSION_LIMIT verb changes the session limit and contention-winner polarities for parallel-sessions. Refer to the *SNA Transaction Programmer's Reference Manual for LU Type 6.2* for details about the CHANGE_SESSION_LIMIT verb.

AS/400 APPC uses the Change Session Maximum (CHGSSNMAX) command as support for this verb. For more information about this command, see page 4-4.

LU_NAME

RMTLOCNAME, LCLLOCNAME, and RMTNETID parameters on the CHGSSNMAX command.

MODE_NAME

MODE parameter on the CHGSSNMAX command.

LU_MODE_SESSION_LIMIT

MAXSSN parameter on the CHGSSNMAX command.

RETURN_CODE

Messages are returned.

INITIALIZE_SESSION_LIMIT: AS/400 APPC uses the Start Mode (STRMOD) command as support for this verb. For more information about this command, see "Start Mode (STRMOD) Command" on page 4-2.

LU_NAME

RMTLOCNAME, RMTNETID, and LCLLOCNAME parameters on the STRMOD command.

MODE_NAME

MODE parameter on the STRMOD command.

MIN_CONWINNERS_SOURCE

LCLCTLSSN parameter on the CRTMODD command.

RETURN_CODE

Messages are returned.

PROCESS_SESSION_LIMIT: This verb processes the session limit, contention-winner polarities, and related change-number-of-sessions (CNOS) parameters from the source LU and, if necessary, negotiates them to values acceptable to the target LU. This verb is used for parallel session connections only.

There are no AS/400 commands or parameters that map to this verb.

RESET_SESSION_LIMIT: This verb resets to 0 the session limit for both single or parallel-session connections, and the contention-winner polarities for the parallel-session connections.

DRAIN_TARGET(NO)

The complete pended requests (CPLPNDRQS) parameter on the ENDMOD command.

Session-Control Verbs

The following verbs are in a subcategory of control operator verbs called the session-control verbs, which are used to activate and deactivate an LU-LU session.

ACTIVATE_SESSION: The ACTIVATE_SESSION verb activates a session with the specified mode name to the target LU. The session is activated as a contention winner for either the source LU or target LU.

There are no AS/400 commands or parameters that map to this verb.

DEACTIVATE_SESSION: The DEACTIVATE_SESSION verb deactivates the specified LU-LU session.

There are no AS/400 commands or parameters that map to this verb.

LU Definition Verbs

The following verbs are in a subcategory of control operator verbs called the LU definition verbs, which are used to define, change, or examine the local operating parameters of the local LU.

DEFINE_LOCAL_LU: The DEFINE_LOCAL_LU defines the fully qualified name for the local LU, and initializes or changes parameters that control the operation of the local LU.

Note: AS/400 system supports multiple local LUs.

AS/400 APPC uses the CRTDEVAPPC command as support for this verb.

FULLY_QUALIFIED_NAME

LCLLOCNAME parameter on the CRTDEVAPPC command, or LCLNETID parameter on the CHGNETA command.

RETURN_CODE

Messages are returned.

DEFINE_MODE: The DEFINE_MODE initializes or changes parameters that control the operation of the local LU in conjunction with a group of sessions to the specified remote LU, the session group being identified by a mode name.

AS/400 APPC uses the CRTMODD command as support for this verb.

MODE_NAME

MODD parameter on the CRTMODD command.

SEND_MAX_RU_SIZE_UPPER_BOUND

MAXLENRU parameter on the CRTMODD command.

RECEIVE_MAX_RU_SIZE_UPPER_BOUND

MAXLENRU parameter on the CRTMODD command.

CONWINNER_AUTO_ACTIVATE_LIMIT

PREESTSSN parameter on the CRTMODD command.

LOCAL_MAX_SESSION_LIMIT

PREESTSSN parameter on the CRTMODD command.

RETURN_CODE

Messages are returned.

DEFINE_REMOTE_LU: The DEFINE_REMOTE_LU initializes or changes parameters that control the operation of the local LU in conjunction with a remote LU.

AS/400 uses the CRTDEVAPPC command as support for this verb.

Note: AS/400 supports multiple local LUs; therefore, the remote LU is defined relative to a particular local LU.

FULLY_QUALIFIED_LU_NAME

RMTLOCNAME parameter on the CRTDEVAPPC command.

PARALLEL_SESSION_SUPPORT

SNGSSN parameter on the CRTDEVAPPC command.

LU_LU_PASSWORD

LOCPWD parameter on the CRTDEVAPPC command.

SECURITY_ACCEPTANCE

SECURELOC parameter on the CRTDEVAPPC command.

RETURN_CODE

Messages are returned.

DISPLAY_LOCAL_LU:

The DISPLAY_LOCAL_LU returns current values of parameters that control the operation of the local LU.

AS/400 APPC uses the DSPDEVD command as support for this verb.

DISPLAY_MODE: The DISPLAY_MODE verb returns current values of parameters that control the operation of the local LU in conjunction with a group of sessions to a remote LU, the session group being identified by a mode name.

AS/400 APPC uses the Display Mode Description (DSPMODD) and the Display Mode Status (DSPMODSTS) commands as support for this verb. For more information about this command, see "Displaying the Mode Status" on page 4-6.

DISPLAY_REMOTE_LU:

The DISPLAY_REMOTE_LU verb returns current values of parameters that control the operation of the local LU in conjunction with a remote LU.

AS/400 APPC uses the DSPDEVD command as support for this verb.

ICF Implementation of the LU Type 6.2 Architecture

This section provides the following information about the ICF implementation of the LU type 6.2 architecture:

- Specifying the resource parameter
- Mapped conversation verbs
- Basic conversation verbs
- Miscellaneous verbs

- Mapping of LU 6.2 return codes to ICF return codes.

Specifying the Resource Parameter

The RESOURCE parameter for LU type 6.2 verbs is specified in ICF by the program device name. The program device name is specified in the PGMDEV parameter of the ADDICFDEVE and OVRICFDEVE commands and in the following ways:

- COBOL/400: The identifier parameter of the ACQUIRE statement or the TERMINAL option of the WRITE and READ statements.
- RPG/400: The program device identifier specified in factor 1 of the calculation specifications.
- C/400: For the _Racquire and _Rrelease functions, the program device name is specified as the second positional parameter. The fread and fwrite functions should be preceded by a _Rpgmdev statement to specify the program device name to be used if multiple sessions are active within the application.
- FORTRAN/400: Does not support multiple sessions for one ICF file. Therefore, program device names are not used. Instead, you must implicitly acquire the session by specifying ACQPGMDEV (program device name) on the CRTICFF, CHGICFF, and OVRICFF commands. You must use a separate file for each ICF session.

Mapped Conversation Verbs

MC_ALLOCATE: There is no single ICF operation that is equivalent to the MC_ALLOCATE verb. The functions of the verb are accomplished by the combination of the acquire operation and the evoke function.

Note: Some of the information used by these operations is specified on commands that run before the application program issues these requests.

LU_NAME

RMTLOCNAME, DEV, RMTNETID, LCLLOCNAME parameters on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands.

MODE_NAME

MODE parameter on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands.

TPN

For data description specifications (DDS), the remote program and library name are specified following the EVOKE function.

For system-supplied formats, the remote program name is in positions 1 through 8 of the data buffer of the source program and the library name is in positions 25 through 32 of the same buffer.

Note: The local program is responsible for ensuring that the remote program and library names are in a format that is acceptable to the remote system naming requirements.

TYPE

(BASIC_CONVERSATION/
MAPPED_CONVERSATION)

To allocate a basic conversation, the CNVTYPE parameter on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands must be specified as *USER.

To allocate a mapped conversation, the CNVTYPE parameter on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands must be specified as *SYS.

RETURN_CONTROL

WHEN_SESSION_ALLOCATED is always used.

- SYNVLV DDS keyword (*NONE or *CONFIRM supported) specified with EVOKE keyword.
- For system-supplied formats, support always defaulted to NONE. System-supplied formats do not support CONFIRM processing.

SYNC_LEVEL(NONE/CONFIRM)

SYNC_POINT is not supported.

- SYNVLV DDS keyword (*NONE or *CONFIRM supported) specified with EVOKE keyword.
- For system-supplied format, support is always defaulted to NONE. System-supplied formats do not support CONFIRM processing.

SECURITY

SECURITY(NONE) is done by not specifying the SECURITY DDS keyword or by specifying SECURITY(3 *NONE).

SECURITY(SAME) is done by specifying SECURITY(3 *USER). If a profile ID was received to start this job, it is passed on whenever an EVOKE is done by specifying SECURITY(3 *USER).

SECURITY(PGM) is done in the following manner:

- DDS SECURITY keyword (user ID, password, and profile ID are all allowed) specified with the EVOKE keyword.
- For system-supplied formats the following can be specified in the source program's data buffer:
 - Positions 9 through 16, password
 - Positions 17 through 24, user ID

Note: For additional information on how to specify security refer to "APPC Security Considerations" on page 3-6.

PIP

For DDS, the parameter data can be specified on the EVOKE keyword.

For system-supplied formats, the parameter data is specified in positions 57-xxxx of the data buffer of the source program.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

MC_CONFIRM: The MC_CONFIRM verb ends a message and asks the remote transaction program to confirm that no errors have been detected in the message. Support for MC_CONFIRM is provided by the CONFIRM DDS keyword. MC_CONFIRM is not supported when using system-supplied formats.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

REQUEST_TO_SEND_RECEIVED

Information is returned in the communications device dependent area of the I/O feedback area as follows:

- Request-to-write indicator
- Minor return code of 10 for major return code of 00

MC_CONFIRMED: The MC_CONFIRMED verb sends a confirmation reply to the remote program. By sending MC_CONFIRMED, the local application program takes responsibility for any data it has received. This verb is only valid in response to an MC_CONFIRM request.

A confirmed response is sent by using the RSPCONFIRM DDS keyword.

RESOURCE

Refer to “Specifying the Resource Parameter” on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

MC_DEALLOCATE: An AS/400 program starts deallocation by issuing an output operation with the detach function specified. If the program is a source program, then an end-of-session function, release, or close operation must follow the detach function to complete the deallocation.

If an AS/400 program receives a return code indicating a detach function was received, then an end-of-session or a close operation may be used to complete the deallocation. The RELEASE or close operation will deallocate the session, only if the AS/400 program is a source program.

RESOURCE

Refer to “Specifying the Resource Parameter” on page C-3.

TYPE(LOCAL)

This value is implicit when issuing an end-of-session function, a release, or close operation after a detach function has been received.

- For DDS, use the EOS keyword.
- For system-supplied formats, use \$\$EOS.

TYPE(SYNC_LEVEL)

The AS/400 system does not support this function for ICF. The SYNLEVEL value can be determined by using the ICF get-attribute operation and checking the synchronization level field in the attribute information. Use either TYPE(CONFIRM) or TYPE(FLUSH).

TYPE(CONFIRM)

Use if conversation was allocated with SYNLEVEL (*CONFIRM).

- For data description specifications (DDS), use the CONFIRM keyword.
- Not supported for system-supplied formats.

TYPE(FLUSH)

Use if conversation was allocated with SYNLEVEL (*NONE).

- For DDS, use the FRCDDTA keyword.
- Not supported for system-supplied formats.

TYPE(ABEND)

An end-of-session function, close operation (for source program), or the application ended without sending or receiving a detach function.

- For DDS, use the EOS keyword.
- For system-supplied formats, use \$\$EOS.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of I/O feedback area

MC_FLUSH: The MC_FLUSH verb causes all buffered data and control information to be sent. MC_FLUSH is done by the AS/400 system following an output operation that specifies the FRCDTA DDS keyword. This function is not available when using system-supplied formats.

RESOURCE

Refer to the "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of I/O feedback area

MC_GET_ATTRIBUTES: The MC_GET_ATTRIBUTES verb requests information about the conversation that is attached to the program. This verb is supported by the get-attribute operation of ICF. Refer to "GET_ATTRIBUTES" on page C-14 for information concerning this operation.

MC_POST_ON_RECEIPT: The MC_POST_ON_RECEIPT verb causes the LU to post the specified conversation when information is available for the program to receive. The information can be data, conversation status, or a request for confirmation. WAIT should be issued after MC_POST_ON_RECEIPT in order to wait for posting to occur.

Note: APPC uses the invite function to perform this verb. There is no length parameter on the ICF invite function. The AS/400 system does not use a length value to determine when to perform posting. The AS/400 system will only perform posting when a record is received or when data is received.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of I/O feedback area

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

MC_PREPARE_TO_RECEIVE: The MC_PREPARE_TO_RECEIVE verb changes the conversation from send state to receive state so that the AS/400 program can receive data. This verb is performed in combination with a SEND_DATA operation when an allow write or invite function is specified on a DDS output format or for a system-supplied format that also specifies an invite (for example, \$\$\$SEND). Note that the invite function also performs the MC_POST_ON_RECEIPT verb.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

TYPE(SYNC_LEVEL)

The AS/400 system does not support this function for ICF. The SYNLVL value can be determined by using the ICF get-attribute operation and checking the Synchronization level field in the attribute information. Use either TYPE(CONFIRM) or TYPE(FLUSH).

TYPE(CONFIRM)

Use if conversation was allocated with SYNLVL(*CONFIRM).

- For data description specifications (DDS), use the CONFIRM keyword.
- Not supported for system-supplied formats.

Should be used if conversation was allocated with SYNLVL (*CONFIRM).

TYPE(FLUSH)

This is done implicitly.

LOCKS(SHORT)

Implicit when CONFIRM is specified.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

MC_RECEIVE_AND_WAIT: The MC_RECEIVE_AND_WAIT verb waits for information to arrive on the specified conversation and then receives the information. If information is already available, the program receives it without waiting. The information can be data, conversation status, or a request for confirma-

tion. To perform an MC_RECEIVE_AND_WAIT, the AS/400 program issues a READ operation.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

LENGTH

Requested length is the length of the record format. Returned length is in the communications device dependent area of the I/O feedback area.

MAP_NAME

The record format used for the READ operation is returned in the communications device dependent area of the I/O feedback area.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

REQUEST_TO_SEND_RECEIVED

Information is returned by the request-to-write indicator in the communication device dependent area of the I/O feedback area.

DATA

Application program's data buffer.

WHAT_RECEIVED

Response indicators and major and minor return codes in communication device dependent area of I/O feedback area supply similar information. Supported values are

- DATA_COMPLETE
- DATA_TRUNCATED
- DATA_INCOMPLETE
- CONTROL_DATA_COMPLETE
- CONTROL_DATA_TRUNCATED
- CONTROL_DATA_INCOMPLETE
- SEND
- CONFIRM
- CONFIRM_SEND
- CONFIRM_DEALLOCATE

Note: The AS/400 system can return more than one value on the same call. For example, DATA_COMPLETE and CONFIRM are indicated by return code 0015 and a RCVCONFIRM DDS response indicator.

Figure C-1 on page C-8 contrasts WHAT_RECEIVED with the major and minor return codes and the response indicators. Note that some major and minor return codes provide more information than WHAT_RECEIVED. For example, return code 0008 is equivalent to DATA_COMPLETE and a RETURN_CODE of DEALLOCATE_NORMAL. DATA_INCOMPLETE is supported when *RETAIN is specified for the OVRFLWDTA parameter.

Figure C-1 (Page 1 of 3). MC_WHAT_RECEIVED Parameters with Corresponding Return Codes and Response Indicators

Major Code 00										
ICF Return Codes	DATA_COMPLETE	DATA_TRUNCATED	DATA_INCOMPLETE	CONTROL_DATA_COMPLETE	CONTROL_DATA_TRUNCATED	CONTROL_DATA_INCOMPLETE	SEND	CONFIRM	CONFIRM_SEND	CONFIRM_DEALLOCATE
0000	X						X			
0001	X									
0004				X			X			
0005				X						
0008 ¹	X									
000C ¹				X						
0010	X									
0014	X								X	
0015	X							X		
001C	X									X
0044				X					X	
0045				X				X		
0046				X						X
Data Received with the Program Start Request (Major Code 00)										
ICF Return Codes	DATA_COMPLETE	DATA_TRUNCATED	DATA_INCOMPLETE	CONTROL_DATA_COMPLETE	CONTROL_DATA_TRUNCATED	CONTROL_DATA_INCOMPLETE	SEND	CONFIRM	CONFIRM_SEND	CONFIRM_DEALLOCATE
0002				X						
0006				X			X			
0011 ¹				X						
0013				X					X	
0018				X				X		
001D				X						X
Major Code 02										
ICF Return Codes	DATA_COMPLETE	DATA_TRUNCATED	DATA_INCOMPLETE	CONTROL_DATA_COMPLETE	CONTROL_DATA_TRUNCATED	CONTROL_DATA_INCOMPLETE	SEND	CONFIRM	CONFIRM_SEND	CONFIRM_DEALLOCATE
0200	X						X			
0201	X									
0204				X			X			
0205				X						
0208 ¹	X									
020C ¹				X						
0214	X								X	
0215	X							X		
021C	X									X
0244				X					X	
0245				X				X		
0246				X						X

¹The LU 6.2 architected return code DEALLOCATE is returned by this ICF return code.

Figure C-1 (Page 2 of 3). MC_WHAT_RECEIVED Parameters with Corresponding Return Codes and Response Indicators

Data Received by the Program Start Request (Major Code 02)										
ICF Return Codes	DATA_COMPLETE	DATA_TRUNCATED	DATA_INCOMPLETE	CONTROL_DATA_COMPLETE	CONTROL_DATA_TRUNCATED	CONTROL_DATA_INCOMPLETE	SEND	CONFIRM	CONFIRM_SEND	CONFIRM_DEALLOCATE
0202				X						
0206				X			X			
0211 ¹				X						
0213				X					X	
0218				X				X		
021D				X						X
Major Code 03										
ICF Return Codes	DATA_COMPLETE	DATA_TRUNCATED	DATA_INCOMPLETE	CONTROL_DATA_COMPLETE	CONTROL_DATA_TRUNCATED	CONTROL_DATA_INCOMPLETE	SEND	CONFIRM	CONFIRM_SEND	CONFIRM_DEALLOCATE
0300							X			
0301										
0304				X			X			
0305				X						
0308 ¹										
030C ¹				X						
0314									X	
0315								X		
031C										X
0344				X					X	
0345				X				X		
0346				X						X
Data Received with the Program Start Request (Major Code 03)										
ICF Return Codes	DATA_COMPLETE	DATA_TRUNCATED	DATA_INCOMPLETE	CONTROL_DATA_COMPLETE	CONTROL_DATA_TRUNCATED	CONTROL_DATA_INCOMPLETE	SEND	CONFIRM	CONFIRM_SEND	CONFIRM_DEALLOCATE
0302				X						
0306				X			X			
0311 ¹				X						
0313				X					X	
0318				X				X		
031D				X						X
Major Code 34										
ICF Return Codes	DATA_COMPLETE	DATA_TRUNCATED	DATA_INCOMPLETE	CONTROL_DATA_COMPLETE	CONTROL_DATA_TRUNCATED	CONTROL_DATA_INCOMPLETE	SEND	CONFIRM	CONFIRM_SEND	CONFIRM_DEALLOCATE
3421					X					
3431		X								
3471			X							
3481						X				

¹The LU 6.2 architected return code DEALLOCATE is returned by this ICF return code.

Figure C-1 (Page 3 of 3). MC_WHAT_RECEIVED Parameters with Corresponding Return Codes and Response Indicators

Data Received with the Program Start Request (Major Code 34)										
ICF Return Codes	DATA_COM- PLETE	DATA_TRUN- CATED	DATA_INCOM- PLETE	CONTROL_ DATA_ COM- PLETE	CONTROL_ DATA_ TRUN- CATED	CONTROL_ DATA_ INCOM- PLETE	SEND	CON- FIRM	CONFIRM_ SEND	CONFIRM_ DEALLO- CATE
3422					X					
Response Indicators										
Response Indicators	CONTROL_ DATA_ COM- PLETE	CONTROL_ DATA_ TRUN- CATED	CONTROL_ DATA_ INCOM- PLETE	SEND	CON- FIRM	CONFIRM_ SEND	CONFIRM_ DEALLO- CATE			
RCVCNFRM					X	X	X			
RCVTRNRND				X		X				
RCVDETACH								X		
RCVCTLDTA	X	X	X							

¹The LU 6.2 architected return code DEALLOCATE is returned by this ICF return code.

MC_RECEIVE_IMMEDIATE: Receives any information that is available from the specified mapped conversation, but does not wait for information to arrive. The information can be data, mapped conversation status, or a request for confirmation. Control is returned to the program with an indication of whether any information was received and, if so, the type of information.

There are no ICF operations or functions that map to this verb.

MC_REQUEST_TO_SEND: The MC_REQUEST_TO_SEND verb tells the remote program that the local program requests to enter send state for the conversation. The conversation is changed to send state when the local program receives a SEND indication from the remote program.

The MC_REQUEST_TO_SEND verb is specified by using a request-to-write function. The request-to-write function is specified in DDS by using the RQSWRT keyword and in system-supplied formats with a \$\$RCD. The DDS format may also include the INVITE keyword. The system-supplied format has an implicit invite specified. Note that the invite function also performs the MC_POST_ON_RECEIPT verb.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

MC_SEND_DATA: The MC_SEND_DATA verb sends data or control information to the remote program. This operation is verb is used with a write operation.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

DATA

Application program's data buffer.

LENGTH

For DDS, implicit by the record format or specified explicitly by the VARLEN keyword.

For system-supplied formats, the first 4 bytes of the output buffer of the application program contain the length of data to send.

MAP_NAME

Indicates that map name processing is to be performed.

- For DDS use the FMTNAME keyword and specify *RMTFMT on the ADDICFDEVE or OVRICFDEVE command.
- Not supported by system-supplied formats.

USER_CONTROL_DATA

Indicates that the data record contains user control data.

- For DDS, use the CTLDTA keyword.
- Not supported by system-supplied formats.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

REQUEST_TO_SEND_RECEIVED

Communications device dependent area of the I/O feedback area as follows:

- Request-to-write indicator
- Minor return code of 10 for major return code of 00

MC_SEND_ERROR: The MC_SEND_ERROR verb informs the remote transaction program that the local program detected an error. The AS/400 system uses the FAIL DDS keyword or the \$\$FAIL system-supplied format to provide this function.

RESOURCE

Refer to the "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

REQUEST_TO_SEND_RECEIVED

Communications device dependent area of the I/O feedback area as follows:

- Request-to-write indicator
- Minor return code of 10 for major return code of 00

MC_TEST: The MC_TEST verb determines if the requested status information is available. When using ICF APPC, the application program can check the request-to-write indicator in the I/O feedback area to determine if a REQUEST_TO_SEND has been received. This field in the I/O feedback area is only updated on

an application I/O boundary. If a REQUEST_TO_SEND is sent to a program after it has issued its last operation, the field will not be updated.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

TEST(Request To Send Received)

Indicator in the communications device dependent area of the I/O feedback area.

TEST(POSTED)

A get-attributes operation. Refer to position 41 in Figure C-2 on page C-14 for more information.

Note: The ICF APPC implementation does not have a function to perform the MC_TEST verb. Because this function does not exist, the AS/400 system does not reset the Request_to_Send_Received indicator value in the same way as the architecture does when the MC_TEST verb is issued.

Basic Conversation Verbs

ALLOCATE: The LU type 6.2 verb ALLOCATE builds a conversation (AS/400 transaction) to a named partner program at another logical unit (LU). For the AS/400 APPC support, it is specified by the combination of the acquire operation and evoke function.

Note: Some of the information used by these operations is specified on commands that run before the application program issues these requests.

LU_NAME

RMTLOCNAME, DEV, RMTNETID, and LCLLOCNAME parameters on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands.

MODE_NAME

MODE parameter on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands.

TPN

For DDS, the remote program and library name are specified following the EVOKE keyword.

For system-supplied formats, the remote program name is in positions 1 through 8 of the data buffer of the source program and the library name is positions 25 through 32 of the same buffer.

TYPE

(BASIC_CONVERSATION/
MAPPED_CONVERSATION)

To allocate a basic conversation, the CNVTYPE parameter on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands must be specified as *USER.

To allocate a mapped conversation, the CNVTYPE parameter on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands must be specified as *SYS.

RETURN_CONTROL

WHEN_SESSION_ALLOCATED is always used. CONVERSATION_GROUP_ID is not supported.

SYNC_LEVEL(NONE/CONFIRM)

SYNC_POINT is not supported.

- SYNVLV DDS keyword (*NONE or

*CONFIRM supported) specified with EVOKE keyword.

- For system-supplied format, support is always defaulted to NONE. System-supplied formats do not support CONFIRM processing.

SECURITY

SECURITY(NONE) is done by not specifying the SECURITY DDS keyword or by specifying SECURITY(3 *NONE).

SECURITY(SAME) is done by specifying SECURITY(3 *USER). If a profile ID was received to start this job, it is passed on whenever an EVOKE is done by specifying SECURITY(3 *USER).

SECURITY(PGM) is done in the following manner:

DDS SECURITY keyword (user ID, password, and profile ID are all allowed) specified with the EVOKE keyword.

For system-supplied formats the following can be specified in the source program's data buffer:

- Positions 9 through 16, password
- Positions 17 through 24, user ID

Note: For additional information on how to specify security refer to "APPC Security Considerations" on page 3-6.

PIP

For DDS, the parameter data can be specified on the EVOKE keyword.

For system-supplied formats, the parameter data is specified in positions 57-xxxx of the source program's data buffer.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

CONFIRM: CONFIRM ends a message and asks the remote transaction program to confirm that no errors have been detected in the message. Support for CONFIRM is provided by the CONFIRM DDS keyword. CONFIRM is not supported when using system-supplied formats.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

REQUEST_TO_SEND_RECEIVED

Communications device dependent area of feedback area as follows:

- Request-to-write indicator
- Minor return code of 10 for major return code 00

CONFIRMED: CONFIRMED sends a confirmation reply to the remote program. By sending CONFIRMED, the local application program takes responsibility for any data it has received. This verb is only valid in response to a CONFIRM request.

A confirmed response is sent by using a DDS record format with the RSPCONFIRM DDS keyword.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

DEALLOCATE: An AS/400 program starts deallocation by issuing an output operation with detach function. If the program is a source program, then an end-of-session function, release, or close operation must follow the detach function to complete the deallocation.

If an AS/400 program receives a return code indicating a detach function was received, then an end-of-session function or close operation may be used to complete the deallocation. The release operation will deallocate the session, only if the AS/400 program is a source program.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

TYPE(LOCAL)

Implicit by issuing an end-of-session function, release, or close operation after a detach indication has been received.

- For DDS, use the EOS keyword
- For system-supplied formats, use \$\$EOS

TYPE(SYNC_LEVEL)

The AS/400 system does not support this function for ICF. The SYNLEVEL value can be determined by using the ICF get-attribute operation and checking the synchronization level field in the attribute information.

TYPE(CONFIRM)

Use if conversation was allocated with SYNLEVEL (*CONFIRM).

- For data description specifications (DDS), use the CONFIRM keyword.
- Not supported for system-supplied formats.

TYPE(FLUSH)

This is done implicitly for ICF. Use if conversation was allocated with SYNLEVEL (*NONE).

- For DDS, use the DETACH keyword.
- For system-supplied formats, use \$SENDET.

TYPE(ABEND_PROG)

An end-of-session function, a close operation (for source program), or the application ended without sending or receiving a detach indication.

- For DDS, use the EOS keyword.
- For system-supplied formats, use \$\$EOS.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of I/O feedback area

FLUSH: The FLUSH verb causes all buffered data and control information to be sent. FLUSH is requested by using the FRCDDTA DDS keyword. This function is not available when using system-supplied formats.

RESOURCE

Refer to the “Specifying the Resource Parameter” on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of I/O feedback area

GET_ATTRIBUTES: The GET_ATTRIBUTES verb requests information about the conversation that is attached to the program. This verb is supported by the get-attribute ICF operation. The values are returned to your record buffer; the length of the record buffer must be at least 240 bytes. The values are returned in the order specified in Figure C-2.

Figure C-2 (Page 1 of 3). Attribute Information Fields

Position	Value	Meaning
1 through 10	Name	Program device name: The name the program used to identify the program device in the file it will read and write from.
11 through 20	Name	Device description name: The device description associated with the program device name (specified during configuration and optionally on the ADDICFDEVE or OVRICFDEVE command).
21 through 30	Name	User ID: If the program was started locally, this is the user ID used to sign on the work station. If the program was started as a result of a program start request, this is the user ID used to start the target program.
31	I	The device is an ICF device type.
	D	The device is a display device.
	U	Unknown.
32 through 37	APPC	APPC communications type.
38	Y	This is a requesting program device.
	N	This is a session acquired by a source program.
39	Y	Program device has been acquired.
	N	Program device has not been acquired.
40	Y	Input is invited for this program device.
	N	Input is not invited for this program device.
41	Y	Invited input is available for this program device.
	N	Invited input is not available for this program device.
42 through 50	Reserved	Not applicable to communications.
51	Y	Session has an active transaction.
	N	Session does not have an active transaction.

Figure C-2 (Page 2 of 3). Attribute Information Fields

Position	Value	Meaning
52 ¹	0	Synchronization level is NONE.
	1	Synchronization level is CONFIRM.
53	M	Mapped conversation.
	B	Basic conversation.
54 through 61	Name	Remote location name: This is the remote location associated with the program device name (specified during configuration and on the ADDICFDEVE or OVRICFDEVE command).
62 through 69	Name	Local logical unit (LU) name.
70 through 77	Name	Local network ID.
78 through 85 ²	Name	Remote LU name.
86 through 93 ²	Name	Remote network ID.
94 through 101 ³	Name	Mode: This is the mode associated with the program device name (specified during configuration and optionally on the ADDICFDEVE or OVRICFDEVE command).
102 through 144	Reserved	
145 through 146	Binary	ISDN remote number length in bytes, including type and plan.
147 through 148	00	ISDN unknown remote number type.
	01	ISDN international remote number type.
	02	ISDN national remote number type.
	03	ISDN network specific remote number type.
	04	ISDN subscriber remote number type.
	06	ISDN abbreviated remote number type.
	149 through 150	00
149 through 150	01	ISDN/telephony remote number plan.
	03	ISDN data remote number plan.
	04	ISDN Telex remote number plan.
	08	ISDN national standard remote number plan.
	09	ISDN private remote number plan.
151 through 154	Reserved	
155 through 194	Character	ISDN remote number (blank padded EBCDIC).
195 through 196	Binary	ISDN remote subaddress length in bytes, including type.
197 through 198	00	ISDN NSAP remote subaddress type.
	02	ISDN user defined remote subaddress type.
199 through 238	Character	ISDN remote subaddress (0 padded hexadecimal).
239	Reserved	
240	0	Incoming ISDN call.
	1	Outgoing ISDN call.
	2	Non-ISDN connection.
241 through 242	Binary	X.25 remote network address length.
243 through 274	Character	X.25 remote network address.
275 through 278	Reserved	
279 through 280	Binary	X.25 remote address extension length.

Figure C-2 (Page 3 of 3). Attribute Information Fields

Position	Value	Meaning
281	0	Address assigned according to ISO 8348/AD2.
	2	Not an ISO 8348/AD2 type of address.
282 through 321	Character	Remote address extension.
322 through 325	Reserved	
326	0	Incoming X.25 switched virtual circuit (SVC).
	1	Outgoing X.25 SVC.
	2	Not X.25 SVC.
327 through 390	Character	Evoked transaction program name.

1 This is the SYNC_LEVEL.

2 Remote LU name is the PARTNER_LU_NAME. Remote network ID, when concatenated by a period (.) with the remote LU name, is the PARTNER_FULLY_QUALIFIED_LU_NAME.

3 This is the MODE_NAME.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of I/O feedback area

RESOURCE

Refer to “Specifying the Resource Parameter” on page C-3.

POST_ON_RECEIPT:

The POST_ON_RECEIPT verb causes the LU to post the specified conversation when information is available for the program to receive. The information can be data, conversation status, or a request for confirmation. WAIT should be issued after POST_ON_RECEIPT in order to wait for posting to occur.

APPC uses the invite function to perform this verb.

Note: There is no length parameter on the ICF invite function. The AS/400 system does not use a length value to determine when to perform posting. The AS/400 system will only perform posting when a complete LL is received or when information other than data is received.

RESOURCE

Refer to “Specifying the Resource Parameter” on page C-3.

FILL(LL)

This is the only supported value.

PREPARE_TO_RECEIVE:

The PREPARE_TO_RECEIVE verb changes the conversation from send state to receive state so that the program can receive data. This verb is performed in combination with a SEND_DATA operation when an allow write or invite function is specified on a DDS output format, or for a system-supplied format that also specifies an invite (for example, \$\$\$SEND). Note that the invite function also performs the POST_ON_RECEIPT verb.

RESOURCE

Refer to “Specifying the Resource Parameter” on page C-3.

TYPE(SYNC_LEVEL)

The AS/400 system does not support this function for ICF. The SYNVLV value can be determined by using the ICF get-attribute operation and checking the synchronization level field in the attribute information. Use either TYPE(CONFIRM) or TYPE(FLUSH).

TYPE(CONFIRM)

Use if conversation was allocated with SYNVLV (*CONFIRM).

- For DDS, use the CONFIRM keyword.
- Not supported for system-supplied formats.

TYPE(FLUSH)

Use if conversation was allocated with SYNVLV (*NONE).

LOCKS(SHORT)

Implicit when CONFIRM is specified.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

RECEIVE_AND_WAIT: The RECEIVE_AND_WAIT verb waits for information to arrive on the specified conversation and then receives the information. If information is already available, the program receives it without waiting. The information can be data, conversation status, or a request for confirmation. To perform a RECEIVE_AND_WAIT the AS/400 program issues a READ operation to a specific program device name.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

FILL(LL)

VARBUFMTG DDS keyword not specified.
Implicit when using system-supplied formats.

FILL(BUFFER)

VARBUFMTG DDS keyword specified. Not supported when using system-supplied formats.

LENGTH

Requested length is the length of the record format area. Returned length is located in the first 2 bytes of the input buffer and in the communications device dependent area of the I/O feedback area.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

REQUEST_TO_SEND_RECEIVED

Information is returned by the request-to-write indicator in the communication device dependent area of the I/O feedback area.

DATA

Application program's data buffer.

WHAT_RECEIVED

Response indicators and major and minor return codes in communication device dependent area of I/O feedback area. For more information on this parameter, refer to "MC_RECEIVE_AND_WAIT" on page C-6.

The tables in Figure C-3 on page C-18 and Figure C-4 on page C-18 contrast WHAT_RECEIVED with the major and minor return codes and the response indicators. Note that some major and minor return codes provide more information than WHAT_RECEIVED. For example, return code 0008 is equivalent to DATA_COMPLETE and a RETURN_CODE of DEALLOCATE_NORMAL. DATA_INCOMPLETE is supported when *RETAIN is specified for the OVRFLWDTA parameter.

Figure C-3. WHAT_RECEIVED Parameters and Corresponding Return Codes and Response Indicators

WHAT_RECEIVED Parameters	Corresponding ICF Return Codes												
	0000	0001	0008 ³	0010	0014	0015	001C ³	0200	0201	0208 ³	0214	0215	021C ³
DATA ¹	X	X	X	X	X	X	X	X	X	X	X	X	X
DATA_COMPLETE ²	X	X	X	X	X	X	X	X	X	X	X	X	X
LL_TRUNCATED ²													
SEND	X							X					
CONFIRM						X						X	
CONFIRM_SEND					X						X		
CONFIRM_DEALLOCATE							X						X

¹ DATA is returned only when FILL(BUFFER) is used with VARBUFMTG.
² DATA_COMPLETE and LL_TRUNCATED are returned only when FILL(LL) is used.
³ The LU 6.2 architected return code DEALLOCATE is returned by this ICF return code.

Figure C-4. WHAT_RECEIVED Parameters and Corresponding Return Codes and Response Indicators

WHAT_RECEIVED Parameters	Corresponding ICF Return Codes							Response Indicators (DDS Keywords)		
	0300	0301	0314	0315	031C ³	3431	3471	RCVCONFIRM	RCVTRNRND	RCVDETACH
DATA ¹		X								
DATA_COMPLETE ²		X								
LL_TRUNCATED ²						X				
DATA_INCOMPLETE							X			
SEND	X								X	
CONFIRM				X				X		
CONFIRM_SEND			X					X	X	
CONFIRM_DEALLOCATE					X			X		X

¹ DATA is returned only when FILL(BUFFER) is used with VARBUFMTG.
² DATA_COMPLETE and LL_TRUNCATED are returned only when FILL(LL) is used.
³ The LU 6.2 architected return code DEALLOCATE is returned by this ICF return code.

RECEIVE_IMMEDIATE: Receives any information that is available from the specified conversation, but does not wait for information to arrive. The information can be data, conversation status, or a request for confirmation. Control is returned to the program with an indication of whether any information was received and, if so, the type of information.

There are no ICF operations or functions that map to this verb.

REQUEST_TO_SEND: The REQUEST_TO_SEND verb tells the remote program that the local program requests to enter send state for the conversation. The conversation is changed to send state when the local program receives a SEND indication from the remote program.

The REQUEST_TO_SEND verb is specified with a request-to-write function. The REQUEST WRITE is specified on a DDS specification using the RQSWRT keyword and in system-supplied formats with a \$\$RCD. The DDS format may

also include the INVITE keyword. The system-supplied format has an implicit invite specified. Note that the invite function also performs the POST_ON_RECEIPT verb.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

SEND_DATA: The SEND_DATA verb sends data or control information to the remote program. Done with a write operation of a format with a data length greater than zero.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

DATA

Application program's data buffer.

LENGTH

For DDS with the VARBUFMGMT keyword specified, implicit by the record format or specified explicitly by the VARLEN keyword.

For DDS without the VARBUFMGMT keyword specified, the first 2 bytes in the output buffer of the application program contain the length of data to send.

For system-supplied formats, the first 4 bytes in the output buffer of the application program contain the length of data to send.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

REQUEST_TO_SEND_RECEIVED

Communications device dependent area of the I/O feedback area as follows:

- Request-to-write indicator
- Minor return code of 10 for major return code of 00

SEND_ERROR: The SEND_ERROR verb informs the remote transaction program that the local program detected an error. Use the FAIL DDS keyword or the \$\$FAIL system-supplied format to provide this function.

RESOURCE

Refer to the "Specifying the Resource Parameter" on page C-3.

TYPE(PROG)

Implicit. This is the only supported value.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

REQUEST_TO_SEND_RECEIVED

Communications device dependent area of the I/O feedback area as follows:

- Request-to-write indicator

- Minor return code of 10 with major return code of 00

TEST: The TEST verb determines if the requested status information is available. When using ICF APPC, the application program can check the request-to-write indicator in the I/O feedback area to determine if a REQUEST_TO_SEND has been received. This field in the I/O feedback area is only updated on an application I/O boundary. If a REQUEST_TO_SEND is sent to a program after it has issued its last operation, the field will not be updated.

RESOURCE

Refer to "Specifying the Resource Parameter" on page C-3.

TEST(REQUEST_TO_SEND_RECEIVED)

Indicator in the communications device dependent area of the I/O feedback area.

TEST(POSTED)

A get-attributes operation. Refer to position 41 in Figure C-2 on page C-14 for more information.

Note: The ICF APPC implementation does not have a function to perform the TEST verb. Because this function does not exist, the AS/400 system does not reset the Request_to_Send_Received indicator value in the same way as the architecture does when the TEST verb is issued.

Miscellaneous Verbs

GET_TP_PROPERTIES: The GET_TP_PROPERTIES verb is used to return information pertaining to the local transaction program. This verb is supported by the get-attribute operation of ICF.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

OWN_FULLY_QUALIFIED_LU_NAME

On the AS/400 system, this is the local network ID concatenated by a period (.) with the local LU name. The local LU name is returned in positions 62 through 69, and the

local network ID is returned in positions 70 through 77 of the data returned by ICF get-attribute operation. Refer to Figure C-2 on page C-14 for more information.

SECURITY_USER_ID

This is returned in positions 21 through 30 of the data returned by the ICF get-attribute operation. Refer to Figure C-2 on page C-14 for more information.

GET_TYPE: The GET_TYPE verb is used to determine the conversation type. This verb is supported by the get-attribute operation of ICF.

RESOURCE

Refer to “Specifying the Resource Parameter” on page C-3.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

TYPE

The conversation type is returned in position 53 of the data returned. Refer to Figure C-2 on page C-14 for more information.

WAIT: The WAIT verb waits for data or information from one or more active conversations. APPC uses the read-from-invited-program-devices operation to wait for input from one or more previously invited conversations.

RESOURCE_LIST

Implied by the previous invites.

RETURN_CODE

Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

RESOURCE_POSTED

Returned in the I/O feedback area.

Mapping of LU 6.2 Return Codes to ICF Return Codes

Figure C-5 provides a mapping between the LU 6.2 return codes and the ICF return codes. The values (or subcodes) for the LU 6.2 return codes listed in the table are passed to a program after the LU has completed the processing of a verb. The return code indicates the result of the processing and can be passed on any verb that has a RETURN_CODE parameter. For detailed descriptions of the LU 6.2 return codes and subcodes, refer to the *SNA Transaction Programmer's Reference Manual for LU Type 6.2*.

Also provided is a brief description of the ICF return codes. For details of the ICF return code descriptions, refer to Appendix B.

Note: Because the return code mapping is not one-to-one, you should note that this table can only provide a general guide as to how return codes correspond. For example, the ICF return code 80C0 can correlate to any of the following LU 6.2 return codes:

- DEALLOCATE_ABEND
- PRODUCT_SPECIFIC_ERROR
- RESOURCE_FAILURE_NO_RETRY
- RESOURCE_FAILURE_RETRY.

Figure C-5 (Page 1 of 3). Mapping of LU 6.2 Return Codes to ICF Return Codes

LU 6.2 Return Code or Subcode	ICF Return Code	Description	
ALLOCATION_FAILURE_NO_RETRY	80EB	Open not successful.	
	80ED	Record format level mismatch occurred, or temporary file close option not allowed for this file type.	
	80F8	File already open or marked in error on a previous return code.	
	8281	System error condition detected.	
	8282	Device is not usable.	
	82A6	BIND command was not successful.	
	82EA	Format selection *RECID cannot be used with the ICF file.	
	82EC	CNVTYPE(*USER) does not support FMTSLT(*RMTFMT).	
	82EE	Device not supported, or your program is trying to acquire the requesting program device in a program that was not started by a program start request.	
	82F0	Error in ICF file.	
	82F2	CNVTYPE specification not valid on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.	
	ALLOCATION_FAILURE_RETRY	80B3	ICF file in use.
		80EB	Open operation not successful.
80ED		Record format level mismatch occurred, or temporary file close option not allowed for this file type.	
81C2		Session not available.	
82A8		Maximum number of program devices allowed for ICF file reached.	
82AB		Device description not varied on.	
82B3		No sessions currently available in the specified communications configuration.	
CONVERSATION_TYPE_MISMATCH	83CF	Either your program or the remote program does not support the specified conversation type.	
DEALLOCATE_ABEND	80C0	Abnormal end of session or remote protocol error.	
	8197	Remote system ended transmission abnormally, session ended.	
	8327	No active transaction.	
DEALLOCATE_ABEND_PROG	8197	Abnormal end of session (program).	
	83F1	Abnormal end of session (program).	
DEALLOCATE_ABEND_SVC	81C5	Abnormal end of session (service).	
DEALLOCATE_ABEND_TIMER	81C6	Abnormal end of session (timer).	
DEALLOCATE_NORMAL	0008	Detach indication received with data.	
	0308	Detach indication received without data.	
MAP_EXECUTION_FAILURE	No mapping.		
MAP_NOT_FOUND	83E0	Record format was not defined for the ICF file.	
MAPPING_NOT_SUPPORTED	No mapping.		
OK	Major return codes 00, 02, 03, or 34	Normal completion.	

Figure C-5 (Page 2 of 3). Mapping of LU 6.2 Return Codes to ICF Return Codes

LU 6.2 Return Code or Subcode	ICF Return Code	Description
PARAMETER_ERROR	82AA	Remote location name not found.
	82C3	Mode specified not valid.
	8334	Program name not specified correctly.
	831E	Operation not valid, or a combination of operations that was not valid was specified.
PIP_NOT_ALLOWED	83D1	PIP data not allowed by remote program.
PIP_NOT_SPECIFIED_CORRECTLY	83D2	PIP data not specified correctly for remote program.
POSTING_NOT_ACTIVE	1100	The read-from-invited-program-devices operation was not successful because your program tried this operation when no program devices were invited and no timer function was in effect.
PROG_ERROR_NO_TRUNC	83C7	Your program received a fail indication (TYPE= PROG) with no data. No data was truncated.
PROG_ERROR_PURGING	83C9	Your program received a fail indication (TYPE= PROG) with or without a confirm indication. Data may have been lost.
PROG_ERROR_TRUNC	83CB	Your program received a fail indication (TYPE= PROG). The last logical record has been truncated.
PROGRAM_PARAMETER_CHECK	8233	Program device name not valid.
	831F	Program specified data or length not valid.
	83CD	The input or output operation issued by your program was not successful because your program attempted a confirm operation while it was still in receive state, or because a confirm operation was specified for a transaction that was started with a synchronization level of *CONFIRM.
	83F3	Length specification (LL) not valid.
PROGRAM_STATE_CHECK	0412	Output not allowed.
	0800	Cannot acquire a session that is already acquired.
	82A9	Acquire to *REQUESTER device failed.
	830B	Session not active.
	8327	No active transaction.
	8329	An evoke function that was not valid was detected in this session.
	832C	A release operation following an invite function was detected.
	832D	Following an invite function, your program issued a request-to-write indication or an additional invite function.
	832F	Transaction already active.
	83CD	Input or output operation was not successful.
83D5	Confirm request in progress.	
83D6	RSPCONFIRM function not valid.	
83F9	Length (LL) truncation error.	

Figure C-5 (Page 3 of 3). Mapping of LU 6.2 Return Codes to ICF Return Codes

LU 6.2 Return Code or Subcode	ICF Return Code	Description
RESOURCE_FAILURE_NO_RETRY	8082	Device is not usable.
	80C0	Session ended abnormally or remote protocol error.
	80EB	Open operation not successful.
	8196	Session was ended locally.
	8197	Remote system ended transmission abnormally, session ended.
RESOURCE_FAILURE_RETRY	8081	System error detected.
	80C0	Abnormal end of session or remote protocol error.
	8191	Permanent line or controller error.
	8196	Session was ended locally.
	8197	Remote system ended transmission abnormally, session ended.
	81E9	Data received does not match any record formats in file.
SECURITY_NOT_VALID	83F8	Program device in error.
	80EF	Open operation not authorized to user.
	82EF	Acquire to device not authorized to user, or device is in service mode.
SYNC_LEVEL_NOT_SUPPORTED_BY_PGM	83CE	Security information specified not valid.
	83D3	Synchronization level not valid.
SVC_ERROR_NO_TRUNC	83C8	Your program received a fail indication (TYPE=SVC) with no data. No data was truncated.
SVC_ERROR_PURGING	83CA	Your program received a fail indication (TYPE=SVC) with or without a confirm indication. Data may have been lost.
SVC_ERROR_TRUNC	83CC	Your program received a fail indication (TYPE=SVC). The last logical record has been truncated.
TPN_NOT_RECOGNIZED	8316	Target program could not be found.
TP_NOT_AVAIL_NO_RETRY	80D0	Program specified on evoke function not available; program cannot issue another evoke function.
TP_NOT_AVAIL_RETRY	83D0	Cannot start program specified on the evoke function; program can try again.
UNSUCCESSFUL	No mapping.	

For information on return code values for mapped conversation verbs using the WHAT_RECEIVED parameter, see Figure C-1 on page C-8. For information on return code values for basic conversation verbs using the WHAT_RECEIVED parameter, see Figure C-3 on page C-18 and Figure C-4 on page C-18.

LU 6.2 Conversation Verb Option Sets Used by the AS/400 System

This section provides a table of the conversation verb option sets that the AS/400 system sup-

ports. An LU 6.2 product may provide support for all verbs or a permitted subset of them. The

permitted subset for LU 6.2 products is defined by means of a base set and a number of option sets. By using the option set numbering, any LU 6.2 product implementation can precisely identify its support of LU 6.2 functions.

Base and option sets for the verbs also depend on local and remote support. **Local support** refers to information flowing only within a local LU or flowing from a local LU to a remote LU; **remote support** refers to information flowing from a remote LU to a local LU. In the following

table, the option set description provides information about the local and remote support for the verbs that are in an option set.

Notes:

1. Figure C-6 includes only those functions that are *not* in the base option set. All other verbs that the AS/400 system supports and are described in this appendix are assumed to be in the base option set.
2. Brackets [] indicate that the contents are optional.

Figure C-6 (Page 1 of 3). Supported Option Sets

Set Number	Set Name	Set Description
101	Flush the LU's send buffer	This option set allows a program to explicitly cause the LU to flush its send buffer. This option set includes the [MC_]FLUSH verb, but just for local support, because the remote support for this verb is in the LU 6.2 base set of functions.
102	Get attributes	This option set allows a program to obtain attributes of a mapped conversation. This option set includes the MC_GET_ATTRIBUTES verb; in contrast, the GET_ATTRIBUTES verb for basic conversations is part of the LU 6.2 base set of functions.
104	Post on receipt with wait	This option set allows a program to request posting of multiple conversations and then to wait (suspend its processing) until information is available on any one of the conversations. This option set includes the [MC_]POST_ON_RECEIPT verb and the WAIT verb, which is a type-independent verb. Option set 105 is a prerequisite.
105	Prepare to receive	This option set allows a program to change the conversation from send state to receive state and at the same time flush the LU's send buffer, request confirmation, or request sync point. This option set includes the [MC_]PREPARE_TO_RECEIVE verb, but just for local support, because the remote support for this verb is in the LU 6.2 base set of functions.
106	Receive immediate	This option set allows a program to receive whatever information is available on a conversation without having to request posting of the conversation. This option set includes the [MC_]RECEIVE_IMMEDIATE verb. Option set 105 is a prerequisite.
110	Get conversation type	Note: Supported only by CPI Communications. This option set allows a program that supports both the basic conversation and mapped conversation protocol boundaries to determine which category of verbs it should use in conjunction with a resource_ID. This option set includes the GET_TYPE verb, a type-independent verb.

Figure C-6 (Page 2 of 3). Supported Option Sets

Set Number	Set Name	Set Description
203	Immediate allocation of a session	<p>This option set allows a program to allocate a contention-winner session only if one is immediately available; otherwise, the allocation is not successful. This option set relates to the RETURN_CONTROL parameter of the [MC_]ALLOCATE verb.</p> <p>Note: Supported only by CPI Communications.</p>
204	Conversations between programs located at the same LU	<p>This option set allows a local program to allocate a conversation to a program located at the same LU as the local program. This option set relates to the LU_NAME parameter of the [MC_]ALLOCATE verb.</p>
211	Session-level LU-LU verification	<p>This option set allows a program or an operator to designate the LU-LU passwords, associated with remote LUs, that the local LU uses to verify the identity of a remote LU at session activation time. This option set relates to the LU_LU_PASSWORD of the DEFINE_REMOTE_LU verb. This option set is a prerequisite for every other security-related option set.</p>
212	User ID verification	<p>This option set allows a program or an operator to designate the user IDs and associated passwords that the local LU uses to verify the identity of a user ID carried on allocation requests it receives, and to designate the remote LUs that are permitted to send to the local LU allocation requests carrying a user ID and either a password or an already-verified indication. This option set also allows the program allocating a conversation to specify that the allocation request carry the user ID received on the request that started the program, together with an already-verified indication. This option set relates to the SECURITY parameter of the [MC_]ALLOCATE verb. Option set 211 is a prerequisite.</p>
213	Program-supplied user ID and password	<p>This option set allows the program allocating a conversation to supply the user ID and password to be sent on the allocation request. This option set relates to the SECURITY parameter of the [MC_]ALLOCATE verb. Option set 211 is a prerequisite.</p>
214	User ID authorization	<p>This option set allows a program or an operator to designate the user IDs that are authorized access to specific resources of the LU, such as transaction programs. This option set relates to the SECURITY parameter of the [MC_]ALLOCATE verb. Option set 211 is a prerequisite.</p>
217	Profile pass-through	<p>This option set allows the program allocating a conversation to specify that the allocation request carry the profile received on the request that started the program. This option set relates to the SECURITY parameter of the [MC_]ALLOCATE verb. Option set 211 is a prerequisite.</p>
218	Program-supplied profile	<p>This option set allows the program allocating a conversation to supply the profile to be sent on the allocation request. This option set relates to the SECURITY parameter of the [MC_]ALLOCATE verb. Option set 213 is a prerequisite.</p>

Figure C-6 (Page 3 of 3). Supported Option Sets

Set Number	Set Name	Set Description
241	Send PIP data	This option set allows the local program allocating a conversation to provide the remote program with initialization parameters. This option set relates to the PIP parameter of the [MC_]ALLOCATE verb.
242	Receive PIP data	This option set allows the local program to receive from the remote program allocating a conversation some initialization parameters. This option set relates to the PIP parameter of the [MC_]ALLOCATE verb at the remote LU. Note: C/400 allows PIP data to be received, but does not allow any way to control how much source the program should provide.
243	Accounting	This option set allows an LU implementation to generate and send both a logical-unit-of-work (LUW) identifier and a conversation correlator (CC) to the remote LU. This option set provides the LUW and CC identifiers to be used, along with other information, for accounting purposes. This option set relates to the CONVERSATION_CORRELATOR and LUW_IDENTIFIER parameters of the [MC_]GET_ATTRIBUTES verb.
245	Test for request-to-send received	This option set allows a program to test whether a request-to-send notification has been received on a conversation, for example, following sync point processing. This option set relates to the TEST parameter of the [MC_]TEST verb. Note: Supported only by CPI Communications.
246	Data mapping	This option set allows a program using mapped conversations to request mapping of the data by the local and remote LUs. This option set relates to the MAP_NAME parameter of the MC_SEND_DATA and MC_RECEIVE_AND_WAIT verbs.
247	User control data	This option set allows programs to send and receive data records containing user control data. The user control data has meaning only to the application programs. This option set relates, for mapped conversations only, to the USER_CONTROL_DATA parameter of the MC_SEND_DATA and DISPLAY_TP verbs.
290	Logging of data in a system log	This option set allows a program to record error information in the system's error log. This option set can be used on basic conversations only and relates to the LOG_DATA parameter of the SEND_ERROR and DEALLOCATE verbs. Note: Supported only by CPI Communications.

LU 6.2 Control-Operator Verb Option Sets Used by the AS/400 System

This section provides a table of the control-operator verb option sets that the AS/400 system supports.

Figure C-7 (Page 1 of 2). Supported Option Sets

Set Number	Set Name	Set Description
501	CHANGE_SESSION_LIMIT verb	This option set allows a program or an operator at the source LU to request a change in the (LU,mode) session limit from one nonzero value to another, or a change in the minimum number of contention-winner sessions for the source LU or target LU. This option set includes the CHANGE_SESSION_LIMIT verb, but just for local support because the remote support for this verb is in the LU 6.2 base set of functions.
505	LU-definition verbs	This option set allows a program or an operator to specify the operating parameters of its LU. This option set includes the DEFINE_LOCAL_LU, DEFINE_MODE, DEFINE_REMOTE_LU, DISPLAY_LOCAL_LU, DISPLAY_MODE, and DISPLAY_REMOTE_LU verbs. Note: The AS/400 implementation of this option set does not provide support for DEFINE_TP.
603	DRAIN_TARGET(NO) parameter	This option set allows a program or an operator at the source LU to prevent the target LU from draining its allocation requests as a result of resetting the (LU,mode) session limit to 0. This option set relates to the DRAIN parameter of the RESET_SESSION_LIMIT verb.
605	LU-LU session limit	This option set allows a program or an operator to specify the LU-LU session limit. This option set relates to the LU_SESSION_LIMIT parameter of the DEFINE_LOCAL_LU verb.
606	Locally-known LU names	This option set allows a program or an operator to specify the locally-known names of remote LUs. This option set relates to the LOCALLY_KNOWN_LU_NAME parameter of the DEFINE_REMOTE_LU and DISPLAY_REMOTE_LU verbs. The locally-known LU name may be used in the LU_NAME parameter of the [MC_]ALLOCATE verb and other verbs having the LU_NAME parameter.
610	Maximum RU size bounds	This option set allows a program or an operator to specify the lower and upper bounds for the maximum RU sizes on sessions within an (LU,mode) group. This option set relates to the following: <ul style="list-style-type: none"> • SEND_MAX_RU_SIZE_LOWER_BOUND • SEND_MAX_RU_SIZE_UPPER_BOUND • RECEIVE_MAX_RU_SIZE_LOWER_BOUND • RECEIVE_MAX_RU_SIZE_UPPER_BOUND parameters of the DEFINE_MODE and DISPLAY_MODE verbs.
612	Contention winner automatic activation limit	This option set allows a program or an operator to specify the limit for automatically activating contention-winner sessions within an (LU,mode) group. This option set relates to the CONWINNER_AUTO_ACTIVATE_LIMIT parameter of the DEFINE_MODE and DISPLAY_MODE verbs.

Figure C-7 (Page 2 of 2). Supported Option Sets

Set Number	Set Name	Set Description
613	Local maximum (LU,mode) session limit	This option set allows a program or an operator to specify the maximum value that can be used during change-number-of-sessions, or CNOS, processing of the (LU,mode) session limit. This option set relates to the CNOS_MAX_SESSION_LIMIT parameter of the DEFINE_MODE and DISPLAY_MODE verbs.

Appendix D. APPC Configuration Examples

This appendix provides the following APPC configuration examples:

- A two-system APPC network (without APPN, switched)
- A two-system APPC network (without APPN, nonswitched)
- X.21 Short-Hold Mode communications between two AS/400 systems
- Defining controller descriptions for programs communicating on the same system

Configuration Example for a Two-System APPC Network (without APPN, Switched)

In this example, two AS/400 systems (Figure D-1) are being configured to communicate with each other using switched connections without APPN networking functions.

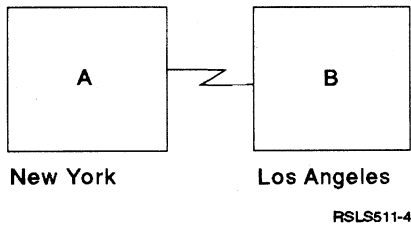


Figure D-1. Two-System APPC Network (without APPN)

In the following example, default values are used for all parameters not explicitly defined. Refer to the description of the commands in this chapter for a description of those parameters that are specifically for APPC; see the *CL Reference* for the complete syntax of the commands and the parameters.

The name assigned to each description created is the same as the name of the destination being defined in that description. For example, the line description configured in New York for the connection to Los Angeles is named LOSANGEL.

Names (such as location names), telephone numbers, exchange identifiers, and other values shown in the examples are for illustration only. The values you assign to your configuration are dependent on your network requirements.

Creating the Line Description (New York to Los Angeles): The line used in this example is an SDLC switched line. The command used to create the line is CRTLINS DLC. The following displays show the creation of this line description.

```

CREATE LINE DESC (SDLC) (CRTLINS DLC)

Type choices, press Enter.

Line description . . . . . > LALINSw      Name
Resource names . . . . . > LING11      Name
+ for more values

Online at IP . . . . . *YES             *YES, *NO
Data link mode . . . . . *NEG           *NEG, *PRI, *SEC
Physical interface . . . . . *RS232V24  *RS232V24, *V35, *X21, ...
Connection type . . . . . > *SWTTP     *NONSWTTP, *SWTTP, *MP, *SHM
Vary on wait . . . . . *NOWAIT        *NOWAIT, 15-180 (1 second)
Autocall unit . . . . . *NC           *NC, *YES
Exchange identifier . . . . . > 05600001  05600000-056FFFF, *SYSGEN
NRZI data encoding . . . . . *YES       *YES, *NO
Line speed . . . . . 9600             600, 1200, 2400, 4800...
Modem type supported . . . . . *NORMAL  *NORMAL, *V54, *IBMWRAP...
Switched connection type . . . . . *BOTM *BOTM, *ANS, *DIA
Autoanswer . . . . . *YES             *YES, *NO
Autodial . . . . . *NO                *NO, *YES

F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys
    
```

```

CREATE LINE DESC (SDLC) (CRTLINS DLC)

Type choices, press Enter.

Calling number . . . . . *NONE
Station address . . . . . 01 2         01-FE
Connect poll retry . . . . . 7         0-64
Maximum frame size . . . . . 521       265, 521, 1033, 2057
Duplex . . . . . *HALF                 *HALF, *FULL
Inactivity timer . . . . . 300         *NOMAX, 150-4200 (0.1 sec)
Poll response delay . . . . . 0        0-2640 (0.0001 seconds)
Nonproductive receive timer . . . . . 320 100-4200 (0.1 seconds)
Idle timer . . . . . 36               5-300 (0.1 seconds)
Connect poll timer . . . . . 36       2-300 (0.1 seconds)
Poll cycle pause . . . . . 0          0-2040 (0.0001 seconds)
Frame retry . . . . . 7              0-64
Data Set Ready drop timer . . . . . 6  3-60 (seconds)
Autoanswer type . . . . . *DTR        *DTR, *CDS*
Remote answer timer . . . . . 60      30, 35, 40, 45 (seconds)...

F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys
    
```

```

CREATE LINE DESC (SDLC) (CRTLINS DLC)

Type choices, press Enter.

Text 'description' . . . . . 'line description for NY to LA
    
```

Considerations for specifying the CRTLINS DLC command:

1 The exchange identifier is used to identify the AS/400 system to the remote system.

2 Valid station addresses are hex 01 to FE.

Creating the Controller Description (New York to Los Angeles):

Because this is an APPC environment (AS/400 system to AS/400 system), the controller is an APPC controller, and the CRTCTLAPPC command is used to define the attributes of the controller. The following displays show the creation of the controller description.

```

CREATE CTL DESC (APPC) (CRTCTLAPPC)

Type choices, press Enter.

Controller description . . . . . > LACT_SW      Name
Link type . . . . . > *SDLC                *IDLC, *LAN, *LOCAL, *SDLC...
Online at IPL . . . . . > *YES              *YES, *NO
Switched connection . . . . . > *YES        *NO, *YES
Short hdlc mode . . . . . > *NO           *NO, *YES
APPN-capable . . . . . > *NO              *YES, *NO
Switched line list . . . . . > LALNSW 1      Name
+ for more values
Maximum frame size . . . . . > *LINKTYPE    265-16393, 265, 521, 1033...
Remote network identifier . . . . . *NETATR    Name, *NETATR, *NONE
Remote control point . . . . .              Name
Exchange identifier . . . . . > 05600002 2      00100000-FFFFFFFF
Initial connection . . . . . > *DIAL        *DIAL, *ANS
Connection number . . . . . > 0135551234 3
Date link role . . . . . > *NEG           *NEG, *PRI, *SEC
Station address . . . . . > 01 4           00-FE
MORE...
F3=Exit  F4=Promot  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys

```

```

CREATE CTL DESC (APPC) (CRTCTLAPPC)

Type choices, press Enter.

Text 'description' . . . . . > 'Controller description NY to LA'

```

Considerations for specifying the CRTCTLAPPC command:

- 1** The switched line list must already exist. This value should match the corresponding line description.
- 2** The exchange identifier is used to identify the remote controller. This parameter is required for switched SDLC lines with APPN(*NO) and no remote control point name specified.
- 3** The connection number is required for APPC controllers with a switched connection unless the initial connection is specified as *ANS.

4 The station address must match the station address specified in the corresponding line description if the data link role is specified as *NEG.

Creating the Device Description (New York to Los Angeles):

Because this is an APPC environment (AS/400 system to AS/400 system), the device is an APPC device, and the CRTDEVAPPC command is used to define the attributes of the device. The following displays show the creation of the APPC device description.

```

CREATE DEVICE DESC (APPC) (CRTDEVAPPC)

Type choices, press Enter.

Device description . . . . . > LADEVSW      Name
Remote location name . . . . . > LOSANGE 1      Name
Online at IPL . . . . . > *YES            *YES, *NO
Local location name . . . . . > NEWYORK 2      Name, *NETATR
Remote network identifier . . . . . *NETATR    Name, *NETATR, *NONE
Attached controller . . . . . > LACT_SW      Name
Mode . . . . . > BLANK 2                 Name, *NETATR
+ for more values > #BADC-
Message queue . . . . . > QSYSOPR          Name, QSYSOPR
Library . . . . . > *LIB                  Name, *LIB, *OURLIB
APPN-capable . . . . . > *NO             *YES, *NO
Single session:
  Single session capable . . . . . *NO        *NO, *YES
  Number of conversations . . . . .          1-512
Location password . . . . . > *NONE
Secure location . . . . . > *NO           *NO, *YES
MORE...
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys

```

```

CREATE DEVICE DESC (APPC) (CRTDEVAPPC)

Type choices, press Enter.

Text 'description' . . . . . > 'Device description NY to LA'

```

Considerations for specifying the CRTDEVAPPC command:

- 1** The remote location name specifies the remote location with which your system will be communicating. This value must match the local location name specified on the remote system's device description.
- 2** The local location name is the unique name by which the AS/400 system is known. This value must match the remote location name on the remote system's device description.

Configuring System B (Los Angeles)

The following example program shows the CL commands used to define the configuration for the system identified as LOSANGEL (System B in Figure D-2 on page D-3). The example shows the commands used in a CL program; the configuration can also be performed using the configuration displays shown previously.

```

/*****/
/*
/* MODULE: LANYAPP          LIBRARY: PUBSCFGS          */
/*
/* LANGUAGE: CL          */
/*
/* FUNCTION: CONFIGURES APPC NODES AS FOLLOWS:      */
/*
/* THIS IS: LOSANGEL TO NEWYORK (switched)          */
/*
/*
/*
/*
/*
/*****/
PGM
/*
/* LOSANGEL TO NEWYORK (switched)                    */
/*****/
/* Create line description for LOSANGEL to NEWYORK */
CRTLINSDLC LIND(NYLINSW) RSRcname(LIN011) +
          CNN(*SWTPP) EXCHID(05600002) +
          STNADR(01)
/* Create controller description for LOSANGEL to NEWYORK */
CRTCTLAPPC CTLD(NYCTLSW) LINKTYPE(*SDLC) SWITCHED(*YES) +
          SWTLINLST(NYLINSW) EXCHID(05600001) +
          INLCNN(*DIAL) CNNBR('4317773333') APPN(*NO) +
          STNADR(01)
/* Create device description for LOSANGEL to NEWYORK */
CRTDEVAPPC DEVD(NYDEVSW) LOCADR(00) RMTLOCNAME(NEWYORK) +
          LCLLOCNAME(LOSANGEL) APPN(*NO) +
          CTL(NYCTLSW) MODE(BLANK #BATCH)
ENDPGM

```

Configuration Example for a Two-System APPC Network (without APPN, Nonswitched)

The commands shown here are those used to configure the AS/400 system for the example shown in Figure D-2 (and Figure 2-1 on page 2-4). In this example, two AS/400 systems are being configured to communicate with each other without using APPN networking functions.

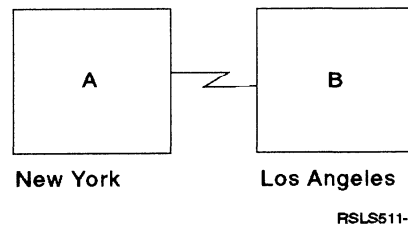


Figure D-2. Two-System APPC Network (without APPN)

In the following example, default values are used for all parameters not explicitly defined. Refer to the description of the commands in this chapter for a description of those parameters that are

specifically for APPC; see the *CL Reference* for the complete syntax of the commands and the parameters.

The name assigned to each description created is the same as the name of the destination being defined in that description. For example, the line description configured in New York for the connection to Los Angeles is named LOSANGEL.

Names (such as location names), telephone numbers, exchange identifiers, and other values

shown in the examples are for illustration only. The values you assign to your configuration are dependent on your network requirements.

Configuring System A (New York): The following CL commands are used to define the configuration for the system identified as NEWYORK (System A in Figure D-1 on page D-1). The example shows the commands as used within a CL program; the configuration can also be performed using the configuration menus.

```

/*****/
/*                                          */
/* MODULE: NYLAAPPC                LIBRARY: PUBSCFGS          */
/*                                          */
/* LANGUAGE: CL                      */
/*                                          */
/* FUNCTION: CONFIGURES APPC NODES AS FOLLOWS:              */
/*                                          */
/*          THIS IS NEWYORK TO LOSANGEL (nonswitched)        */
/*                                          */
/*****/
PGM
/*****/
/*          NEWYORK TO LOSANGEL (nonswitched)                */
/*****/
/* Create line description for NEWYORK to LOSANGEL */
CRTLINS DLC LIND(LOSANGEL) RSRNAME(LIN011)
/* Create controller description for NEWYORK to LOSANGEL */
CRTCTLAPPC CTLD(LOSANGEL) LINKTYPE(*SDLC) APPN(*NO)
          LINE(LOSANGEL) RMTNETID(*NONE) STNADR(01)
/* Create device description for NEWYORK to LOSANGEL */
CRTDEVAPPC DEVD(LOSANGEL) LOCADR(00) RMTLOCNAME(LOSANGEL)
          LCLLOCNAME(NEWYORK) APPN(*NO)
          CTL(LOSANGEL) MODE(BLANK #BATCH)

ENDPGM

```

Creating the Line Description (New York to Los Angeles): The line used in this example is an SDLC nonswitched line. The command used to create the line is Create Line Description (SDLC) (CRTLINS DLC). The parameters specified are:

LIND(LOSANGEL)

The name assigned to the line description is LOSANGEL.

RSRNAME(LIN011)

Specifies that the physical communications port named LIN011 is being defined.

Creating the Controller Description (New York to Los Angeles): Because this is an APPC environment (AS/400 system to AS/400 system), the controller is an APPC controller and the CRTCTLAPPC command is used to define the attributes of the controller. The following attributes are defined by the CRTCTLAPPC command in the example:

CTLD(LOSANGEL)

The name assigned to the controller description is LOSANGEL.

LINKTYPE(*SDLC)

Because this controller is to be attached through an SDLC communications line, the value specified is *SDLC. This value must correspond to the type of line being used as

defined by a create line description command.

APPN(*NO)

This example does not need or use the APPN networking capabilities of the AS/400 system, so APPN(*NO) is specified. All devices attached to this controller must also specify APPN(*NO).

LINE(LOSANGEL)

Specifies the name (LOSANGEL) of the line description to which this controller is attached. This value must match a name specified by the LIND parameter in a line description.

RMTNETID(*NONE)

Because APPN(*NO) is specified, there is no need to define a remote network identifier.

STNADR(01)

The address assigned to the remote controller is hex 01.

Creating the Device Description (New York to Los Angeles):

Because this is an APPC environment (AS/400 system to AS/400 system), the device is an APPC device and the CRTDEVAPPC command is used to define the attributes of the device. The following attributes are defined by the example command:

DEV(LOSANGEL)

Specifies that the name assigned to the device description is LOSANGEL.

LOCADR(00)

The location address should always be specified as hex 00 when the device is associated with an APPC controller.

RMTLOCNAME(LOSANGEL)

Specifies that the remote location name associated with this device description is LOSANGEL.

This value matches the value specified for the LCLLOCNAME parameter at the other system (LOSANGEL).

LCLLOCNAME(NEWYORK)

Specifies the name assigned to the local location, which is NEWYORK in the example.

This value matches the value specified for the RMTLOCNAME parameter at the other system (LOSANGEL).

APPN(*NO)

Specifies that the networking support is not used.

CTL(LOSANGEL)

Specifies that the device description is to be attached to the controller description named LOSANGEL.

MODE(BLANK #BATCH)

Specifies that this device will use either of two modes: BLANK, which is a mode name of all blanks (hex 40), or #BATCH. Both these modes are supplied by IBM. Note that the other location must also use one of these modes when communicating with this location.

Configuring System B (Los Angeles):

The following CL commands are used to define the configuration for the system identified as LOSANGEL. The example shows the commands as used within a CL program; the configuration can also be performed using the configuration menus.

```

/*****
/*
/* MODULE: LANYAPPC LIBRARY: PUBSCFGS
/*
/* LANGUAGE: CL
/*
/* FUNCTION: CONFIGURES APPC NODES AS FOLLOWS:
/*
/*
/* THIS IS LOSANGEL TO NEWYORK (nonswitched)
/*
/*
/*****
PGM
/*****
/* LOSANGEL TO NEWYORK (nonswitched)
/*****
/* Create line description for LOSANGEL to NEWYORK */
CRTLINS DLC LIND(NEWYORK) RSRNAME(LIN012)
/* Create controller description for LOSANGEL to NEWYORK */
CRTCTLAPPC CTLD(NEWYORK) LINKTYPE(*SDLC) APPN(*NO)
LINE(NEWYORK) RMTNETID(*NONE) STNADR(01)
/* Create device description for LOSANGEL to NEWYORK */
CRTDEVAPPC DEVD(NEWYORK) LOCADR(00) RMTLOCNAME(NEWYORK)
LCLLOCNAME(LOSANGEL) APPN(*NO)
CTL(NEWYORK) MODE(BLANK #BATCH)
ENDPGM

```

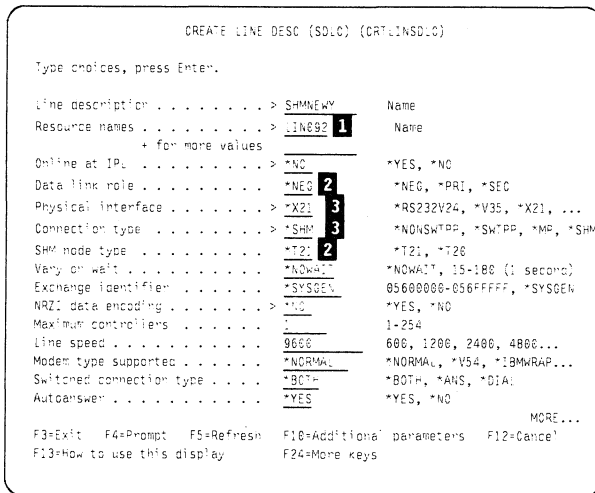
Configuration Example for X.21 Short-Hold Mode

The following example shows a configuration used for X.21 SHM communications between two AS/400 systems, NEWYORK and LOSANGEL.

Configuring the New York System

The following prompt displays show the configuration created for the first AS/400 system, NEWYORK.

Creating the Line Description




```

CREATE LINE DESC (SDLC) (CRTCLNSDLC)

Type choices, press Enter.

Autodial . . . . . > *YES          *NO, *YES
Dial command type . . . . . *NONE      *NONE, *V25BIS
SHM call timer . . . . . 5           *NONE, 1-60 (minutes)
SHM maximum connect timer . . . . . 8       *NOMAX, 1-254 (seconds)
SHM answer delay timer . . . . . 11        *NOMAX, 1-254 (0.1 seconds)
SHM call format . . . . . *DNIC 4     *DNIC, *DCC
SHM access code . . . . . 17 5        Character value
Calling number . . . . . 31019764522 6
Short timer . . . . . 50             10-600 (0.1 seconds)
Long timer . . . . . 600            100-6000 (0.1 seconds)
Short retry . . . . . 7             0-254
Long retry . . . . . 1             0-254
Call progress signal retry . . . . .
+ for more values
Maximum frame size . . . . . 521     265, 521, 1033, 2057
Duplex . . . . . *FULL             *HALF, *FULL
MORE...

F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys

```

Notes:

- 1 *Resource names* (RSRCNAME parameter) are specified to indicate to the system which ports are used for the X.21 SHM line. Because multiple ports are not used in this example, only one resource name is specified for the line descriptions.
- 2 The *Data link role* (ROLE parameter) and *SHM node type* (SHM parameter) must be specified as shown for lines used with APPC controllers.
- 3 *Physical interface* (INTERFACE parameter) and *Connection type* (CNN parameter) must be specified as shown for an X.21 SHM line.
- 4 The *SHM call format* (SHMCALLFMT parameter) is set to *DNIC, indicating that the first 4 digits of the *Calling number* (CALLNBR parameter) represent the data network identification code (DNIC) of the local system.
- 5 The *SHM access code* (SHMACC parameter) specifies the prefix or access code that is attached in front of the international data number.
- 6 The *Calling number* (CALLNBR parameter) represents the connection number of the AS/400 system. For line descriptions that specify SHMNODE(*T21), the calling number must include the DNIC or DCC. In this example, the system will not use the DNIC

when placing calls between the two AS/400 systems because both systems have the same DNIC.

Creating the Controller Description

```

CREATE CTL DESC (APPC) (CRTCTLAPPC)

Type choices, press Enter.

Controller description . . . . . > LOSANGELES Name
Link type . . . . . > *SDLC          *IDLC, *LAN, *LOCAL, *SDLC...
Online at IPL . . . . . > *YES        *YES, *NO
Switched connection . . . . . > *YES 1  *NO, *YES
Short hold mode . . . . . > *YES 1    *NO, *YES
APPC-capable . . . . . > *NO         *YES, *NO
Switched line list . . . . . > SHMNEWY 2 Name
+ for more values
Maximum frame size . . . . . *LINKTYPE 265-16393, 265, 521, 1033...
Remote network identifier . . . . . *NETATR Name, *NETATR, *NONE
Remote control point . . . . . > RCHAS04S 3 Name
Exchange identifier . . . . . 00100000-FFFFFFFF
Initial connection . . . . . *DIAL     *DIAL, *ANS
Connection number . . . . . > 31012991073 4
Data link role . . . . . *NEG        *NEG, *PRI, *SEC
SHM disconnect limit . . . . . 10     1-254, *NOMAX
MORE...

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys

```

```

CREATE CTL DESC (APPC) (CRTCTLAPPC)

Type choices, press Enter.

SHM disconnect timer . . . . . 50      2-3000 (0.1 seconds)
Station address . . . . . C1          00-FE
Text description . . . . . *APPC controller for SHM line to LOSANGELES

```

Notes:

- 1 Both *Switched connection* (SWITCHED parameter) and *Short hold mode* (SHM parameter) must be specified as *YES for X.21 SHM controller descriptions.
- 2 The *Switched line list* (SWTLINLST parameter) must include the name of the line description to which the controller is attached (LIND parameter on the CRTCLNSDLC command).
- 3 The *Remote control point name* (RMTCPNAME parameter) must match the LCLNETID and LCLCPNAME network attributes specified at the remote system.
- 4 The *Connection number* (CNNNBR parameter) specifies the connection number for the remote system. This value must match that specified for the line description *Calling number* (CALLNBR parameter).

Creating the Device Description

```

CREATE DEVICE DESC (APPC) (CRTDEVAPPC)

Type choices, press Enter.

Device description . . . . . > LOSANGE1      Name
Remote location name . . . . . > LOSANGE1 1    Name
Online at IPL . . . . . > *NO              *YES, *NO
Local location name . . . . . > NEWYORK 1    Name, *NETATR
Remote network identifier . . . . . > *NETATR  Name, *NETATR, *NONE
Attached controller . . . . . > LOSANGE1 2    Name
Mode . . . . . > *NETATR                  Name, *NETATR
+ for more values
Message queue . . . . . > QSYSOPR        Name, QSYSOPR
Library . . . . . > *LIB                Name, *LIB., *CURLIB
APPN-capable . . . . . > *NO          *YES, *NO
Single session:
Single session capable . . . . . > *NO      *NO, *YES
Number of conversations . . . . . > *NONE   1-512
Location password . . . . . > *NO
Secure location . . . . . > *NO      *NO, *YES
MORE...
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys
  
```

Notes:

1

The *Remote location name* (RMTLOCNAME parameter) and *Local location name* (LCLLOCNAME parameter) must each match the opposite parameter on the remote system: The RMTLOCNAME value specified for this system (NEWYORK) must match the LCLLOCNAME specified in the device description for the LOSANGEL system.

2

The *Attached controller* (CTL parameter) must specify the name of the controller description to which the device is attached (CTLD parameter on the CRTCTLAPPC command).

Configuring the Los Angeles System

The following prompt displays show the configuration created for the second AS/400 system, LOSANGEL. Considerations for specifying the CRTLNSDLC, CRTCTLAPPC, and CRTDEVAPPC commands are the same as for the NEWYORK system.

Creating the Line Description

```

CREATE LINE DESC (SDLC) (CRTLNSDLC)

Type choices, press Enter.

Line description . . . . . > SHML0SA      Name
Resource names . . . . . > LIN051        Name
+ for more values
Online at IPL . . . . . > *NO          *YES, *NO
Data link role . . . . . > *NEG        *NEG, *PRI, *SEC
Physical interface . . . . . > *X21     *RS232V24, *V35, *X21, ...
Connection type . . . . . > *SMP       *NONSWAPP, *SWTDP, *MP, *SHM
SHM node type . . . . . > *T21     *T21, *T20
Vary on wait . . . . . > *NOWAIT    *NOWAIT, 15-186 (1 second)
Exchange identifier . . . . . > *SYSDEL  05600000-056FFFFF, *SYSGEN
NRZI data encoding . . . . . > *NO      *YES, *NO
Maximum controllers . . . . . > 1        1-254
Line speed . . . . . > 9600         600, 1200, 2400, 4800...
Mode type supported . . . . . > *NORMAL *NORMAL, *V54, *IBMWRAP...
Switched connection type . . . . . > *BOT*  *BOT*, *ANS, *DIAL
Autoswapper . . . . . > *YES       *YES, *NO
MORE...
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys
  
```

```

CREATE LINE DESC (SDLC) (CRTLNSDLC)

Type choices, press Enter.

Autodial . . . . . > *YES          *NO, *YES
Dial command type . . . . . > *NONE     *NONE, *V25B1S
SHM call timer . . . . . > 5          *NONE, 1-60 (minutes)
SHM maximum connect timer . . . . . > 0        *NOMAX, 1-254 (seconds)
SHM answer delay timer . . . . . > 11       *NOMAX, 1-254 (0.1 seconds)
SHM call format . . . . . > *DNC      *DNC, *DCC
SHM access code . . . . . > 17        Character value
Calling number . . . . . > 310:2981873
Short timer . . . . . > 56          10-600 (0.1 seconds)
Long timer . . . . . > 600         100-6000 (0.1 seconds)
Short retry . . . . . > 7          0-254
Long retry . . . . . > 1          0-254
Call progress signal retry . . . . . > *CP54, *CP52, *CP53...
+ for more values
Maximum frame size . . . . . > 521        265, 521, 1033, 2057
Duplex . . . . . > *FULL        *HALF, *FULL
MORE...
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys
  
```

Creating the Controller Description

```

CREATE CTL DESC (APPC) (CRTCTLAPPC)

Type choices, press Enter.

Controller description . . . . . > NEWYORK      Name
Link type . . . . . > *SDLC        *IDL, *LAA, *LOCAL, *SDLC...
Online at IPL . . . . . > *NO          *YES, *NO
Switched connection . . . . . > *YES     *NO, *YES
Short hold mode . . . . . > *YES     *NO, *YES
APPN-capable . . . . . > *NO          *YES, *NO
Switched line list . . . . . > SHML0SA      Name
+ for more values
Maximum frame size . . . . . > *LINKTYPE  265-16393, 265, 521, 1033...
Remote network identifier . . . . . > *NETATR  Name, *NETATR, *NONE
Remote control point . . . . . > RCHAS320  Name
Exchange identifier . . . . . > 00100000-FFFFFFF
Initial connection . . . . . > *DIAL    *DIAL, *ANS
Connection number . . . . . > 31019764522
Data link role . . . . . > *NEG        *NEG, *PRI, *SEC
SHM disconnect limit . . . . . > 10        1-254, *NOMAX
MORE...
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
  
```

```

CREATE CTL DESC (APPC) (CRTCTLAPPC)

Type choices, press Enter.

SHV disconnect timer . . . . . 50          2-3000 (0.1 seconds)
Station address . . . . . C1             00-FE
Text 'description' . . . . . 'APPC controller for NEWYORK system'

```

Creating the Device Description

```

CREATE DEVICE DESC (APPC) (CRTDEVAPPC)

Type choices, press Enter.

Device description . . . . . > NEWYORK      Name
Remote location name . . . . . > NEWYORK    Name
Online at IP1 . . . . . > *NO              *YES, *NO
Local location name . . . . . > LOSANGELES Name, *NETATR
Remote network identifier . . . . . > *NETATR Name, *NETATR, *NONE
Attached controller . . . . . > NEWYORK     Name
Mode . . . . . > *NETATR                   Name, *NETATR
+ for more values
Message queue . . . . . > QSYSOPR          Name, QSYSOPR
library . . . . . > *LIB.                  Name, *LIB., *CURLIB
APPN-capable . . . . . > *NO              *YES, *NO
Single session:
  Single session capable . . . . . > *NO    *NO, *YES
  Number of conversations . . . . . >      1-512
  Location password . . . . . > *NONE
  Secure location . . . . . > *NO          *NO, *YES
MORE...

F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys

```

Configuration Example for Programs Communicating on the Same System

Using a configuration with a link type of *LOCAL is beneficial for developing and debugging application programs before they are used to communicate with a remote system over a communications line.

The following example program shows the CL commands used to define the configuration with a link type of *LOCAL.

```

/*****
/* Create controller and devices for LINKTYPE(*LOCAL) */
/*****
CRTCTLAPPC CTLD(T8189CTL) LINKTYPE(*LOCAL) ONLINE(*NO) +
TEXT('Controller description for APPC +
examples')

CRTDEVAPPC DEVD(T8189DEV1) RMTLOCNAME(T8189LA) +
ONLINE(*NO) LCLLOCNAME(T8189NY) +
RMTNETID(*NETATR) CTL(T8189CTL) +
APPN(*NO) SECURELOC(*YES) +
TEXT('Device description for APPC examples')

CRTDEVAPPC DEVD(T8189DEV2) RMTLOCNAME(T8189NY) +
ONLINE(*NO) LCLLOCNAME(T8189LA) +
RMTNETID(*NETATR) CTL(T8189CTL) +
APPN(*NO) SECURELOC(*YES) +
TEXT('Device description for APPC examples')

```


Appendix E. Example Programs for Applications Using the ICF Application Program Interface

This appendix provides example programs which demonstrate how to use the APPC support on the AS/400 system using ICF. The example programs included in this appendix are also available in the QUSRTOOL library (see file QATTINFO, member T8189INF in library QUSRTOOL).

The example programs are as follows:

- Example 1 (starting on page E-3) shows two partner C/400 programs.
- Example 2 (starting on page E-21) shows two partner COBOL/400 programs.
- Example 3 (starting on page E-36) shows two partner RPG/400 programs.

Each example contains two programs: the local program, which initiates the transaction, and the remote program, which performs services relating to the processing of the transaction.

Note: The term **remote program** is used in the following examples to refer to the program with which the local program communicates, even though the remote program may not be on a remote system. Similarly, the term **remote system** may actually be the same system in which the local program resides.

Figure E-1 illustrates the environment in which the example programs run. The local program requests the entry of a part number from the display station. That part number is then transmitted to the partner program, where a database file is searched. If the number is found, the partner program responds with a positive response followed by the requested information. If the part number is not found, the partner program responds with a negative response followed by an error message.

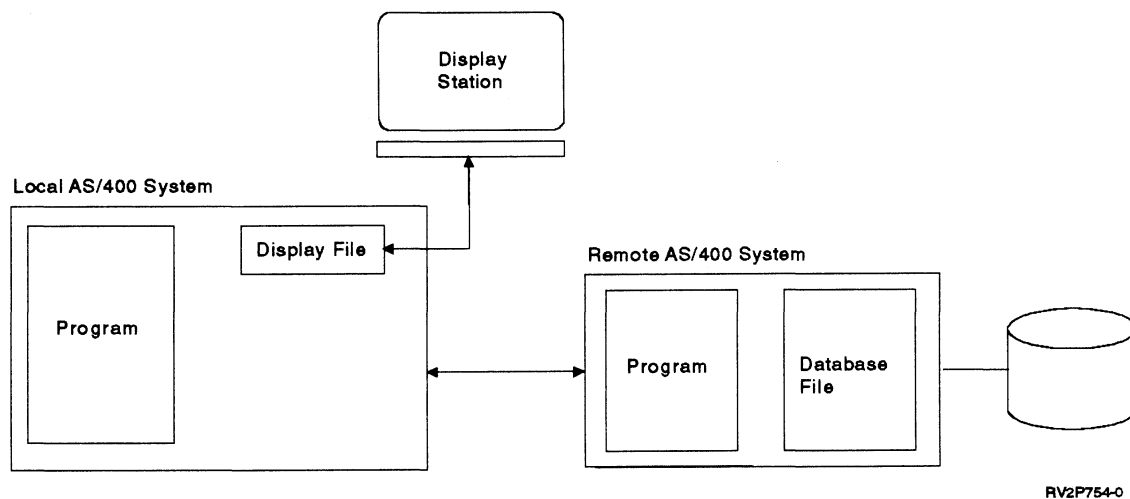


Figure E-1. Inquiry Example

Objects Used by Program Examples

The following objects are used by the program examples:

- ICF file, T8189ICF
- Display file, T8189DSP
- Database file, T8189DB

ICF File Object (T8189ICF)

In this example program an ICF file is used to send records to, and receive records from, the partner program. This file is used by both the local and remote programs. This file was created by using the following command, which must be issued on the local **and** remote systems:

```
CRTRICFF FILE(APPCLIB/T8189ICF) SRCFILE(QUSRTOOL/QATTDDS)
SRCMBR(TC8189) TEXT('ICF file for APPC examples')
```

The command needed to define the program device entry used by the local programs is

```
ADDICFDEVE FILE(APPCLIB/T8189ICF) PGMDEV(ICF00)
           RMTLOCNAME(T8189LA)
```

The command needed to define the program device entry used by the remote programs is

```
ADDICFDEVE FILE(APPCLIB/T8189ICF) PGMDEV(ICF01)
           RMTLOCNAME(*REQUESTER)
```

Note: Even though the ADDICFDEVE command is used, an OVRICFDEVE command could also be used with the same parameters.

The DDS source for the ICF file T8189ICF is shown in Figure E-2.

```
A*****
A*
A*
A*          DDS
A*          FOR THE ICF FILE
A*          USED IN ITEM INQUIRY APPLICATIONS
A*
A*****
A*
A* FILE LEVEL INDICATORS:
A*
A          INDARA
A*
A*****
A*          RECORD FORMATS
A*****
A*
A          R PGMSTR
A
A          EVOKE(&PGMID)
A          SECURITY(3 *USER)
A          SYNVLV(*CONFIRM)
A          PGMID          10A  P
A*
A          R ITEMRQ
A
A          CONFIRM
A          ALWwRT
A          PARTNM          5A
A*
A          R ITEMDS
A
A          ALWwRT
A          PARTDS          25A
A*
A          R ERRDES
A
A          ALWwRT
A          ERROR          40A
A*
A          R PGMEND
A
A          DETACH
A*
A          R EOSREC
A
A          EOS
A*
A          R PGMERR
A
A          FAIL
A*
A          R ITEMOK
A
A          RSPCONFIRM
```

Figure E-2. DDS Source for the ICF File T8189ICF

Display File Object (T8189DSP)

In this example a display file is used by the local program so that a user can enter requests that are to be sent to the remote program. The command used to create the display device file is

```
CRTDSPF FILE(APPCLIB/T8189DSP) SRCFILE(QUSRTOOL/QATTDSS)
        SRCMBR(TD8189) SRCMBR(TD8189)
        TEXT('Display file for APPC examples')
```

The DDS source for the display device file T8189DSP is shown in Figure E-3 on page E-3.

```

A*****
A*
A*          DDS
A*  FOR THE DISPLAY FILE
A*USED IN ITEM INQUIRY APPLICATIONS
A*
A*****
A*
A          DSPSIZ(24 80 *DS3)
A          INDARA
A          CA03(99)
A*
A*****
A*          RECORD FORMATS
A*****
A*
A R PROMPT
A          5 10'Part Number: '
A PARTN  5A B 5 25
A          10 10'Part Description: '
A PARTD  25A 0 10 30
A ERRORL 40A 0 12 10DSPATR(HI)
A          23 5'F3 = Exit'

```

Figure E-3. DDS Source for the Display Device File

Database File Object (T8189DB)

In this example the database file resides on the remote system and contains the part numbers and associated descriptions. The file is used to validate the part number received from the local program. The command used to create the database file (a physical file) is

```

CRTPF FILE(APPCLIB/T8189DB)
SRCFILE(APPCLIB/QATTDSS) SRCMBR(1A8189)
TEXT('Database file for APPC examples')

```

The DDS source for the database file T8189DB is shown in Figure E-4.

```

A*****
A*
A*          DDS
A*  FOR THE DATABASE FILE
A* USED IN ITEM INQUIRY APPLICATIONS
A*
A*****
A*
A          UNIQUE
A R DBRCD
A ITEMNM      5
A ITEMMD      25
A K ITEMNM

```

Figure E-4. DDS Source for the Database File

C/400 Local Program for Inquiry Applications (Example 1)

The following explains the structure of the C/400 local program that sends requests to the partner program for processing.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure E-5 on page E-6.

Note: On any type of error that is not expected (for example, an unexpected ICF return code on an I/O operation), the session is ended and the program ends.

1

This section defines the ICF file (T8189ICF) structures used in the program. T8189ICF is the ICF file used to send records to, and receive records from, the partner program. T8189ICF uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

2

This section defines the display file (T8189DSP) structures used in the program. T8189DSP is the display file used to receive a user's requests and to report the requested information received from the partner program based on the part number specified by the user. T8189DSP uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

3

The internal functions are prototyped so the C/400 compiler knows the type of value returned and the type of parameters passed, if any.

4

The `open_files`, `start_conversation`, and `get_cust_num` functions are called to open files used by the program, start a conversation with the partner program, and obtain the part number to be queried, respectively.

5

The program loops until either F3 is pressed from the work station, which sets the indicator in the indicator area of the display file, or an error occurs in the transaction with the partner program.

6 The part number is sent to the partner program using a write operation. The write operation is issued using the ICF file record format ITEMRO, which contains the confirm (CONFIRM keyword) and allow-write (ALWWRT keyword) functions. When these functions are used, the data is flushed, the data flow direction is changed from send to receive, and a confirmation request is sent to the partner program. The partner program must now respond with a positive or negative response.

7 If the partner program responds with a positive response (ICF return code of 0001) to the confirmation request, a read operation is issued using the ICF file record format ITEMDS to receive the part description. However, if the partner program responds with a negative response (ICF return code of 83C9) to the confirmation request, a read operation is issued using the ICF file record format ERRDES to receive the error message.

8 The `get_cust_num` function is called to display the information returned by the partner program and to obtain the next part number to be queried.

9 The `cleanup` function is called to perform end-of-program processing.

10 The `open_files` function opens the display and ICF files.

11 Separate indicator areas are defined for the files T8189DSP and T8189ICF. The variables `dsp_indic` and `icf_indic` are of the type `_SYSindara`, which is a 99-character array.

12 The separate indicator area for the display file T8189DSP is initialized.

13 The `start_conversation` function establishes a conversation with the partner program.

14 The ICF00 program device is explicitly acquired using the `_Racquire` function. The `acquire-program-device` operation makes the program device available for input or output

operations. A session is implicitly acquired for the work station when T8189DSP is opened.

Note: The program device ICF00 was previously added to the ICF file T8189ICF by the `ADDICFDEVE` command.

15 An evoke request is issued using a write operation. The write operation is issued using the ICF file record format PGMSTR, which contains the `EVOKE`, `SECURITY`, and `SYNLVL` keywords.

Note: On the `EVOKE` keyword, the library name is not specified. If the remote system is an AS/400 system, the library list will be used to search for the program. Also, the remote program that is to be started can be any of the remote programs in this appendix or in Appendix F, "Example Programs for Applications Using the CPI Comm Application Program Interface."

16 The `get_cust_num` function displays the requested information and reads the next number. The part number field will be blank the first time a part number is read.

17 The `check_rc` function determines whether the actual ICF return code received on an operation matches what was expected. If the return codes match, a value of 0 is returned; otherwise, a value of 1 is returned.

Note: Because the I/O feedback areas are updated after each ICF file I/O operation, this function first updates the pointers to the new feedback areas before determining whether the return codes match. Refer to the *ICF Programmer's Guide* for a description of the I/O feedback areas.

18 The `cleanup` function performs end-of-program processing.

19 If an unexpected error was detected and the communications session is still active, a write operation is issued using the ICF file record format EOSREC, which contains the end-of-session (`EOS` keyword) function. The end-of-session function detaches the program from the session.

Note: If the end-of-session function is issued during an active transaction, APPC will end the session abnormally.

20

If no error was detected, then a write operation is issued using the ICF file record format

PGMEND, which contains the detach (DETACH keyword) function. The release operation (_Release function) is then issued to detach the program from the session.

21

The ICF and display files are closed.

```

          * * * * * P R O L O G * * * * *
Program name . . . . . : T8189ICS
Library name . . . . . : APPCLIB
Source file . . . . . : QATTSYSC
Library name . . . . . : QUSRT00L
Source member name . . . . . : T8189ICS
Text Description . . . . . :
Compiler options . . . . . : *SOURCE      *NOXREF      *NOSHOWUSR  *NOSHOWSYS  *NOSHOWSKP  *NOEXPMAC  *NOAGR
                          : *NOPPONLY   *NODEBUG    *GEN        *NOSECLVL  *PRINT      *LOGMSG
Language level options . . . . . : *EXTENDED
Source margins:
Left margin . . . . . : 1
Right margin . . . . . : 32767
Sequence columns:
Left Column . . . . . :
Right Column . . . . . :
Define name . . . . . :
Generation options . . . . . : *NOLIST      *NOXREF      *GEN        *NOATR      *NODUMP      *NOOPTIMIZE *NOALWBND
                          : *NOANNO
Print file . . . . . : QSYSVRT
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
INDEBUB options . . . . . :
Last change . . . . . : 90/09/26 08:26:13
Source description . . . . . :
Compiler . . . . . : IBM C/400 Compiler

```

2

```

          * * * * * S O U R C E * * * * *
Line STMT
1          *...+...1....+...2....+...3....+...4....+...5....+...6....+...7....+...8....+...9.....
2          /******
3          /* Program name.....: T8189ICS
4          /* Program description..: ICF local program
5          /* Language.....: C/400
6          /*
7          /* This program invokes a program to handle part inquiry on
8          /* the remote system. The acquire operation is used to
9          /* establish a communications session. A write operation
10         /* with the evoke function is then issued, which results in
11         /* the establishment of a conversation with the remote
12         /* program. A display which prompts the user for the part
13         /* number for which part information is requested is then
14         /* displayed. When the user presses Enter, a write operation
15         /* is issued (the data sent to the partner program is the
16         /* part number). Note that the write operation was issued
17         /* with the confirm (CONFIRM) and allow-write (ALWVRT)
18         /* functions. These functions results in the flushing of the
19         /* data (to be sent to the partner program), the changing of
20         /* the data flow direction (the partner program can send the
21         /* response), and the sending of a confirmation request to
22         /* the partner program. If the partner program responds
23         /* with a positive response to the confirmation request (using
24         /* the RSPCONFIRM function), the ICF return code on the
25         /* write operation will be set to 0001 (indicating that
26         /* the part number was found); a read operation is then
27         /* issued to receive the part description. However, if
28         /* the partner program responds with a negative response
29         /* to the confirmation request (using the FAIL function),
30         /* the ICF return code on the write operation will be set
31         /* to 83C9 (indicating that the part number was not found);
32         /* a read operation is issued to receive the error message.
33         /*
34         /* The error message or part description (depending on
35         /* whether the part number was found) will be displayed on
36         /* the screen.

```

Figure E-5 (Part 1 of 7). C/400 Inquiry Example — Local Program

```

36      /* */
37      /* This program will continue to handle inquiries until the */
38      /* user presses the F3=Exit key. When F3=Exit is pressed, */
39      /* a write operation with the detach (DETACH) function */
40      /* is issued to end the conversation, and program processing */
41      /* ends. */
42      /* */
43      /* NOTE 1: If an unexpected ICF return code is received on */
44      /* any of the read or write operations, the */
45      /* program will abnormally end the conversation (if */
46      /* it is still active), and program processing will */
47      /* end. */
48      /* */
49      /* NOTE 2: On the receive operation, if the actual received */
50      /* data length (obtained from the I/O feedback area) */
51      /* does not match what was expected, or if the */
52      /* ICF return code is not 0000 (indication that
3
Line STMT
      *...+...1....+...2....+...3....+...4....+...5....+...6....+...7....+...8....+...9.....
53      /* the partner program is ready to receive data), the */
54      /* program will abnormally end the conversation (if */
55      /* it is still active), and program processing will */
56      /* end. */
57      /* */
58      /* NOTE 3: This program can start ANY of the "remote" */
59      /* program examples in the APPC Programmer's */
60      /* Guide by changing the PGMID variable to the */
61      /* remote program that is to be started. */
62      /* */
63      /******
64      |
65      |
66      /******
67      /* Retrieve various structures/utilities that are used in program. */
68      /******
69      #include <stdio.h> /* Standard I/O header */
70      #include <stdlib.h> /* General utilities */
71      #include <string.h> /* String handling utilities */
72      #include <stddef.h> /* Standard definition */
73      #include <xxfdbk.h> /* Feedback area structures */
74      #include <xxasio.h> /* Indicator area structure */
75      #include <recio.h> /* Record i/o routines */
76      |
77      /******
78      /* Define the structures used for reads/writes from/to the ICF file. */
79      /******
80      1 | struct {
81      | char pgmid??(10??); /* target program name */
82      | } pgmstr_i_o = { "T8189ICT " };
83      |
84      | struct {
85      | char partnm??(5??); /* Part number */
86      | } itemrq_i_o;
87      |
88      | struct {
89      | char partds??(25??); /* part description */
90      | } itemds_i_o;
91      |
92      | struct {
93      | char errord??(40??); /* error record */
94      | } errdes_i_o;
95      | char blank40??(40??) = " ";
96      |

```

Figure E-5 (Part 2 of 7). C/400 Inquiry Example – Local Program

```

97 |
98 | /*****
99 | /* Define the structures used for reads/writes from/to the display */
100 | /* file. */
101 | /*****
102 | 2 | struct {
103 | |   char partn??(5??);           /* part number           */
104 | |   char partd??(25??);         /* part description     */
105 | |   char errorl??(40??);        /* error record         */
106 | | } prompt_i_o = { "      ", "
4
Line STMT
107 | *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
108 | | " " " " };
109 | /*****
110 | /* Define constants/flags used in program. */
111 | /*****
112 | #define ERROR 1                /* error during I/O processing */
113 | #define NOERROR 0
114 | #define MATCH 1                /* ICF return code match indication*/
115 | #define NO_MATCH 0
116 | #define RC_0000 0              /* ICF return codes */
117 | #define RC_0001 1
118 | #define RC_83C9 2
119 | |
120 | /*****
121 | /* Declare global variables/functions. */
122 | /*****
123 | 3 | int check_rc(int);
124 | | void cleanup(int);
125 | | void open_files(void);
126 | | void get_cust_num(void);
127 | | void start_conversation(void);
128 | |
129 | | _RFILE *icffptr;             /* Pointer to the ICF file */
130 | | _RFILE *dspfp;              /* Pointer to the display file */
131 | | XXIOFB_T *comm_fdbk;        /* IO Feedback for ICF unique info */
132 | | XXIOFB_DSP_ICF_T *dsp_icf_fdbk; /* IO Feedback - display & ICF file*/
133 | | _SYSindara dsp_indic;       /* indicator area for dsp */
134 | | _SYSindara icf_indic;       /* indicator area for ICF */
135 | |
136 | |
137 | /*****
138 | /* START OF PROGRAM */
139 | /*
140 | /* Files are opened, a conversation with the remote program is
141 | /* started, and the part inquiry screen is displayed. Inquiries
142 | /* are handled until the user presses the F3=Exit key, in which case
143 | /* the conversation will be ended and the program will end.
144 | /*
145 | 4 | main()
146 | | {
147 | | 1 | open_files();
148 | | 2 | start_conversation();
149 | |
150 | | 3 | get_cust_num();
151 | |
152 | | 5 | 4 | while (dsp_indic??(98??) != '1')
153 | | | {
154 | | |
155 | | | /*****
156 | | | /* The part number that the user has requested information */
157 | | | /* for is sent to the remote program using the write */
158 | | | /* operation with the confirm and allow-write functions. */
159 | | | /*****
160 | | | 5 | strncpy(itemrq_i_o.partnm, prompt_i_o.partn, 5);

```

Figure E-5 (Part 3 of 7). C/400 Inquiry Example – Local Program

5

```

Line  STMT
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
161   6 | _Rformat(icffptr, "ITEMRQ  ");
162   7 | _Rwrite(icffptr, &itemrq_i_o, sizeof(itemrq_i_o));
163
164
165   | /******
166   | /* The read operation is issued to receive the response          */
167   | /* from the remote program (the response can either be          */
168   | /* an error message or the part description, depending          */
169   | /* on whether the part was found or not).                        */
170   | /******
171   7 | 8 | if (check_rc(RC_0001) == MATCH)
172   |   | {
173   |   |   _Rformat(icffptr, "ITEMDS  ");
174   | 10 |   _Rreadn(icffptr, &itemds_i_o, sizeof(itemds_i_o), __DFT);
175   |
176   |   |   strncpy(prompt_i_o.pardt, itemds_i_o.pardt, 25);
177   | 12 |   strncpy(prompt_i_o.error1, blank40, 40);
178   |   | }
179   |   | else
180   | 13 |   | if (check_rc(RC_83C9) == MATCH)
181   |   |   | {
182   |   |   |   _Rformat(icffptr, "ERRDES  ");
183   | 15 |   |   _Rreadn(icffptr, &errdes_i_o, sizeof(errdes_i_o), __DFT);
184   |   |   |
185   |   |   |   strncpy(prompt_i_o.error1, errdes_i_o.error1, 40);
186   | 17 |   |   |   strncpy(prompt_i_o.pardt, "          ", 25);
187   |   |   |   }
188   |   |   |   else /* Unexpected return code. */
189   | 18 |   |   |   cleanup(ERROR);
190   |
191   | 19 |   |   if (check_rc(RC_0000) == NO_MATCH) /* Read operations ok? */
192   | 20 |   |   |   cleanup(ERROR);
193   |
194   8 | 21 |   |   get_cust_num();
195   |   |   |   } /* end of while */
196   |
197   9 | 22 | cleanup(NOERROR);
198   |
199   |   | } /* end of MAIN routine */
200   |
201   | /******
202   | /* *****
203   | /* *****
204   | /* *          INTERNAL FUNCTIONS          *
205   | /* *****
206   | /* *****
207   | /******
208   |
209   | /******
210   | /* "OPEN_FILES" function
211   | /*
212   | /* This function opens the display and ICF files, and sets
213   | /* indicator areas for each file.
214   | /******

```

6

```

Line  STMT
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
215  10 | void open_files()
216   | {
217   |   if ((dspfptr= Ropen("T8189DSP", "an+ indicators=y riofb=y" ))
218   |   |   == NULL)
219   |   |   {
220   |   |   |   printf("Display file failed to open.\n");
221   |   |   |   exit(ERROR);
222   |   |   }

```

Figure E-5 (Part 4 of 7). C/400 Inquiry Example — Local Program

```

223 |
224 | if ((icffptr=_Ropen("T8189ICF","ar+ indicators=y riofb=y" ))
225 4 |     == NULL)
226 |     {
227 5 |         printf("ICF file failed to open.\n");
228 6 |         exit(ERROR);
229 |     }
230 |
231 11 7 | _Rindara(dspfptr, dsp_indic);
232 8 | _Rindara(icffptr, icf_indic);
233 |
234 12 9 | memset(dsp_indic, '0', 99);      /* Initialize indicator area.    */
235 |                                /* end open_files...        */
236 |
237 | /*****
238 | /* "START_CONVERSATION" function
239 | /*
240 | /* This function establishes a conversation with the remote system. */
241 | /* The program device is acquired, and an evoke request is issued */
242 | /* to start the program at the remote system.
243 | /*
244 13 | void start_conversation()
245 | {
246 14 1 | _Racquire(icffptr, "ICF00 ");
247 2 | if (check_rc(RC_0000) == NO_MATCH)
248 3 |     cleanup(ERROR);
249 |
250 4 | _Rformat(icffptr, "PGMSTR ");
251 15 5 | _Rwrite(icffptr, &pgmstr_i_o, sizeof(pgmstr_i_o));
252 6 | if (check_rc(RC_0000) == NO_MATCH)
253 7 |     cleanup(ERROR);
254 | }                                /* end start_conversation... */
255 |
256 | /*****
257 | /* "GET_CUST_NUM" function
258 | /*
259 | /* Get a customer number from the display.
260 | /*
261 16 | void get_cust_num()
262 | {
263 1 | _Rformat(dspfptr,"PROMPT ");
264 2 | _Rwrite (dspfp, &prompt_i_o, sizeof(prompt_i_o));
265 3 | memset(dsp_indic, '0', 99);
266 4 | _Rreadn (dspfp, &prompt_i_o, sizeof(prompt_i_o), __DFT);
267 | }                                /* end get_cust_num... */
268 |
7
Line STMT
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
269 | /*****
270 | /* "CHECK_RC" function
271 | /*
272 | /* This function compares the actual ICF return code received
273 | /* on an operation with an ICF return code that was expected.
274 | /* If the return codes match, then an indication that the return
275 | /* codes matched is returned; otherwise, an error indication is
276 | /* returned.
277 | /*
278 17 | check_rc(int rc_type)
279 | {
280 1 | comm_fdbk = _Riobk(icffptr);
281 | dsp_icf_fdbk = (XXIOFB_DSP_ICF_I *) ((char *) comm_fdbk +
282 2 |     comm_fdbk->file_dep_fb_offset);
283 |

```

Figure E-5 (Part 5 of 7). C/400 Inquiry Example — Local Program

```

284 3 | if (rc_type == RC_0000)
285 |     {
286 |         if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&
287 |             strncmp(dsp_icf_fdbk->minor_ret_code, "00", 2) == 0)
288 |             return(MATCH);
289 |         else
290 |             return(NO_MATCH);
291 |     }
292 | else
293 |     if (rc_type == RC_0001)
294 |     {
295 |         if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&
296 |             strncmp(dsp_icf_fdbk->minor_ret_code, "01", 2) == 0)
297 |             return(MATCH);
298 |         else
299 |             return(NO_MATCH);
300 |     }
301 | else
302 |     if (rc_type == RC_83C9)
303 |     {
304 |         if (strncmp(dsp_icf_fdbk->major_ret_code, "83", 2) == 0 &&
305 |             strncmp(dsp_icf_fdbk->minor_ret_code, "C9", 2) == 0)
306 |             return(MATCH);
307 |         else
308 |             return(NO_MATCH);
309 |     }
310 | else
311 |     return(NO_MATCH);
312 | } /* end check_rc... */
313 |
314 |
315 | /*****
316 | /* "CLEANUP" function. */
317 | /*
318 | /* The following code handles the end-of-program processing. */
319 | /* This includes the ending of the conversation with */
320 | /* the remote system (if conversation is active) by either */
321 | /* issuing a write operation with the detach function */
322 | /* followed by a release operation (for non-error conditions), */
8
Line STMT
      *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
323 | /* or by issuing a write operation with the end-of-session */
324 | /* function (for error conditions). */
325 | /*****
326 | 18 void cleanup(int endtype)
327 | |{
328 | |
329 | 19 | if ((endtype == ERROR) &&
330 | |     (strncmp(dsp_icf_fdbk->major_ret_code, "80", 2) != 0) &&
331 | |     (strncmp(dsp_icf_fdbk->major_ret_code, "81", 2) != 0) &&
332 | |     (strncmp(dsp_icf_fdbk->major_ret_code, "82", 2) != 0))
333 | |     {
334 | |         2 | _Rformat(icffptr, "EOSREC ");
335 | |         3 | _Rwrite(icffptr, 0, 0);
336 | |     }
337 | | else
338 | |     {
339 | 20 | 4 | _Rformat(icffptr, "PGMEND ");
340 | |     5 | _Rwrite(icffptr, 0, 0);
341 | |
342 | |     6 | _Rrelease(icffptr, "ICF00 ");
343 | |     }
344 | | }

```

Figure E-5 (Part 6 of 7). C/400 Inquiry Example – Local Program

```

345 |
346 21 7 | _Rclose (icffptr);
347 8 | _Rclose (dspfpftr);
348 |
349 9 | exit(endtype);
350 |}
351 |
/* end cleanup... */

***** END OF SOURCE *****
***** INCLUDES *****
INCNO Include Name Last change Actual Include Name
1 stdio.h 90/09/12 14:42:23 QCC/H/STDIO
2 stddef.h 90/09/12 14:42:22 QCC/H/STDDEF
3 errno.h 90/09/12 14:42:18 QCC/H/ERRNO
4 signal.h 90/09/12 14:42:22 QCC/H/SIGNAL
5 ctype.h 90/09/12 14:42:18 QCC/H/CTYPE
6 stdarg.h 90/09/12 14:42:22 QCC/H/STDARG
7 stdlib.h 90/09/12 14:42:24 QCC/H/STDLIB
8 string.h 90/09/12 14:42:24 QCC/H/STRING
9 stddef.h 90/09/12 14:42:22 QCC/H/STDDEF
10 xxfdbk.h 90/09/12 14:42:28 QCC/H/XXFDBK
11 xxasio.h 90/09/12 14:42:26 QCC/H/XXASIO
12 recio.h 90/09/12 14:42:21 QCC/H/RECIO
13 xxfdbk.h 90/09/12 14:42:28 QCC/H/XXFDBK
***** END OF INCLUDES *****
***** MESSAGE SUMMARY *****
Total Info(0-4) Warning(5-19) Error(20-29) Severe(30-39) Terminal(40-99)
0 0 0 0 0 0
***** END OF MESSAGE SUMMARY *****
ROUTINE BLOCK NUMBER SCOPE TYPE
<MAIN> 2 LOCAL MAIN-PROGRAM
__memset 131 LOCAL PROCEDURE
__strncmp 149 LOCAL PROCEDURE
__strncpy 151 LOCAL PROCEDURE
QXXACQUIRE 167 LOCAL PROCEDURE
QXXRELEASE 168 LOCAL PROCEDURE
QXXFORMAT 169 LOCAL PROCEDURE
QXXINDARA 170 LOCAL PROCEDURE
__reads 192 LOCAL PROCEDURE
__rwrite 195 LOCAL PROCEDURE
__rfmt 202 LOCAL PROCEDURE
check_rc 209 ENTRY PROCEDURE
cleanup 210 ENTRY PROCEDURE
open_files 211 ENTRY PROCEDURE
get_cust_num 212 ENTRY PROCEDURE
start_conversation
213 ENTRY PROCEDURE
main 220 ENTRY PROCEDURE
***** END OF COMPILATION *****

```

Figure E-5 (Part 7 of 7). C/400 Inquiry Example – Local Program

C/400 Remote Program for Inquiry Applications (Example 1)

The following explains the structure of the C/400 remote program that handles requests sent by the partner program.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure E-6 on page E-15.

Note: On any type of error that is not expected (for example, an unexpected ICF return code on

an I/O operation), the session is ended and the program ends.

1

This structure defines the database file (T8189DB) structures used in the program. T8189DB is the database file used to read the customer records.

2

This section defines the ICF file (T8189ICF) structures used in the program. T8189ICF is the ICF file used to send records to, and receive records from, the partner program. T8189ICF uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

- 3** The internal functions are prototyped so the C/400 compiler knows the type of value returned and the type of parameters passed, if any.
- 4** The `open_files` and `start_conversation` functions are called to open files used by the program and to start a conversation with the partner program, respectively.
- 5** The program loops until there are no more requests to process, or an error occurs in the transaction with the partner program.
- 6** A search of the database file is performed using the part number received from the partner program as the key.
- 7** If the part number is found in the database file, a write operation is issued using the ICF file record format `ITEMOK`, which contains the respond-to-confirm (`RSPCONFIRM` keyword) function. As a result, a positive response to the received confirmation request is sent to the partner program. This is followed by a second write operation using the ICF file record format `ITEMDS`, which contains the requested information to be sent and the allow-write (`ALWWRT` keyword) function. When the allow-write function is used, the data is flushed and the data flow direction is changed.
- 8** If the part number is not found in the database file, a write operation is issued using the ICF file record format `PGMERR`, which contains the fail (`FAIL` keyword) function. As a result, a negative response to the received confirmation request is sent to the partner program. This is followed by a second write operation using the ICF file record format `ERRDES`, which contains the error message to be sent and the allow-write function.
- 9** The `cleanup` function is called to perform end-of-program processing.
- 10** The `open_files` function opens the database and ICF files.
- 11** A separate indicator area is defined for the ICF file `T8189ICF`. The variable `icf_indic` is of the type `_SYSindara`, which is a 99-character array.
- 12** The `start_conversation` function establishes a conversation with the partner program by explicitly acquiring the `ICF01` program device using the `_Racquire` function.
Note: The program device `ICF01` was previously added to the ICF file `T8189ICF` by the `ADDICFDEVE` command.
- 13** The `get_cust_num` function waits for a request from the partner program by issuing a read operation using the ICF file record format `ITEMRQ`.
Note: A transaction is processed if data is received with a turnaround indication and the partner program requested confirmation. This is indicated by the ICF return code `0014`.
- 14** The `check_rc` function determines whether the actual ICF return code received on an operation matches what was expected. If the return codes match, a value of 0 is returned; otherwise, a value of 1 is returned.
Note: Because the I/O feedback areas are updated after each ICF file I/O operation, this function first updates the pointers to the new feedback areas before determining whether the return codes match.
- 15** The `cleanup` function performs end-of-program processing.
- 16** If the ICF return code is `0308`, indicating that a detach was received, a release operation (`QXXRELEASE` function) is issued to detach the program from the session.
- 17** If a detach indication was not received and the communications session is still active, a write operation is issued using the ICF file record format `EOSREC`, which contains the end-of-session (`EOS` keyword) function. The end-of-session function detaches the program from the session.

Note: If the end-of-session function is issued during an active transaction, APPC will end the session abnormally.

18

The ICF and database files are closed.

1

***** PROLOG *****

```

Program name . . . . . : T8189ICT
Library name . . . . . : APPCLIB
Source file . . . . . : QATTSYSC
Library name . . . . . : QUSRTOOL
Source member name . . . . . : T8189ICT
Text Description . . . . . :
Compiler options . . . . . : *SOURCE *NOXREF *NOSHOWUSR *NOSHOWSYS *NOSHOWSKP *NOEXPMAC *NOAGR
: *NOPPONLY *NODEBUG *GEN *NOSECLVL *PRINT *LOGMSG
Language level options . . . . . : *EXTENDED
Source margins:
Left margin . . . . . : 1
Right margin . . . . . : 32767
Sequence columns:
Left Column . . . . . :
Right Column . . . . . :
Define name . . . . . :
Generation options . . . . . : *NOLIST *NOXREF *GEN *NOATR *NODUMP *NOOPTIMIZE *NOALWBND
: *NOANNO
Print file . . . . . : QSYSPT
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 . . . . . :
Last change . . . . . : 90/09/26 08:26:16
Source description . . . . . :
Compiler . . . . . : IBM C/400 Compiler

```

2

***** SOURCE *****

```

Line STMT
1          *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
2          /******
3          /* Program name.....: T8189ICT
4          /* Program description.: ICF remote program
5          /* Language.....: C/400
6          /*
7          /* This program accepts the incoming conversation by issuing
8          /* the acquire operation to acquire the requesting program
9          /* device. A read operation is then issued to receive the
10         /* part number from the remote system. The read operation
11         /* completes with an ICF return code of 0014, indicating
12         /* that data with a turnaround indication was received, and
13         /* that the partner program also requested confirmation. The
14         /* database file T8189DB is searched for the received part
15         /* number. If the part number is found, a write operation
16         /* issued with the respond-to-confirm (RSPCONFIRM) function is
17         /* issued, followed by a write operation containing the
18         /* part description corresponding to the part number
19         /* retrieved from the database file. However, if the part
20         /* number is not found, a write operation with the
21         /* negative-response (FAIL) function is issued, followed by
22         /* a write operation containing an error message describing
23         /* the error. The write operation sending either the part
24         /* description or the error message is issued with the
25         /* allow-write (ALWVRT) function. Using the allow-write
26         /* function results in the flushing of the data and the
27         /* changing of the data flow direction. The partner program
28         /* can send more inquiries.
29         /*
30         /* This program will continue to handle inquiries from the
31         /* partner program until a detach indication is received.
32         /* Then the program ends.
33         /*
34         /* NOTE 1: If an unexpected ICF return code is received on
35         /* any of the read or write operations, the
36         /* program will abnormally end the conversation (if

```

Figure E-6 (Part 1 of 7). C/400 Inquiry Example – Remote Program

```

36  /*      it is still active), and program processing will      */
37  /*      end.                                                  */
38  /*      */                                                    */
39  /* NOTE 2: On the receive operation, if the actual received   */
40  /*      data length (obtained from the I/O feedback area)     */
41  /*      does not match what was expected, or if the          */
42  /*      ICF return code is not 0014 (indication that         */
43  /*      data was received with a turnaround indicator,       */
44  /*      and partner program requested confirmation), the     */
45  /*      program will abnormally end the conversation (if     */
46  /*      it is still active), and program processing will     */
47  /*      end.                                                  */
48  /*      */                                                    */
49  /* NOTE 3: This program can be started by ANY of the         */
50  /*      "local" program examples in the APPC Programmer's    */
51  /*      Guide.                                                */
52  /*      */                                                    */
3  Line STMT
    *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
53  /******
54  |
55  /******
56  /* Retrieve various structures/utilities that are used in program. */
57  /******
58  #include <stdio.h>          /* Standard I/O header          */
59  #include <stdlib.h>        /* General utilities          */
60  #include <string.h>        /* String handling utilities  */
61  #include <stddef.h>        /* Standard definition       */
62  #include <xxfdbk.h>        /* Feedback area structures  */
63  #include <xxasio.h>        /* Indicator area structure   */
64  #include <recio.h>         /* record i/o routines       */
65  |
66  /******
67  /* Define the structure used for reads from the database file. */
68  /******
69  1 | struct {
70  |   char partn??(5??);
71  |   char partd??(25??);
72  | } part_rec;
73  |
74  /******
75  /* Define the structures used for reads/writes from/to the ICF file. */
76  /******
77  2 | struct errrdes {
78  |   char errorord??(40??);
79  | } errrdes_i_o;
80  |
81  | struct {
82  |   char partnm??(5??);
83  | } itemrq;
84  |
85  | struct {
86  |   char partds??(25??);
87  | } itemds;
88  |
89  |
90  /******
91  /* Define constants/flags used in program.                    */
92  /******
93  #define ERROR 1          /* error during I/O processing */
94  #define NOERROR 0
95  #define MATCH 1         /* ICF return code match indication */
96  #define NO_MATCH 0
97  #define MORE_REQUESTS 0 /* More request indicator      */
98  #define NO_REQUESTS 1
99  #define RC_0000 0       /* ICF return codes           */
100 #define RC_0001 1
101 #define RC_0308 2
102 #define RC_0014 3
103 |

```

Figure E-6 (Part 2 of 7). C/400 Inquiry Example – Remote Program

```

104 | /*****
105 | /* Declare global variables/functions.
106 | /*****
4 |
Line STMT
   | *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
107 | 3 | int check_rc(int);
108 | | int get_cust_num(void);
109 | | void cleanup(void);
110 | | void open_files(void);
111 | | void start_conversation(void);
112 | |
113 | | char part_not_found??(40??) =
114 | | | "THE REQUESTED PART WAS NOT FOUND ";
115 | |
116 | | _RFILE *icffptr; /* Pointer to the ICF file */
117 | | _RFILE *dbfptr; /* Pointer to database file. */
118 | |
119 | | XXIOFB_T *comm fdbk; /* IO Feedback for ICF unique info */
120 | | XXIOFB_DSP_ICF_T *dsp_icf_fdbk; /* IO feedback - display & ICF file */
121 | |
122 | | _RIOFB_T *db_fdbk; /* IO Feedback - data base file */
123 | |
124 | | _SYSindara icf_indic; /* indicator area for ICF
125 | | size_t size; /* "size_t" is a synonym for the
126 | | /* type of the value returned by
127 | | /* the "sizeof" operator.
128 | |
129 | |
130 | | /*****
131 | | /* START OF PROGRAM
132 | | /*
133 | | /* Files are opened, a conversation with the remote program is
134 | | /* started, and the part inquiry processing starts. Inquiries
135 | | /* are handled until a detach indication is received.
136 | | /*****
137 | | main()
138 | | {
139 | | 4 | 1 | open_files();
140 | | 2 | start_conversation();
141 | |
142 | | 5 | 3 | while (get_cust_num() != NO_REQUESTS)
143 | | | {
144 | | | /*****
145 | | | /* A search of the database file is done using the part
146 | | | /* number as the key.
147 | | | /*****
148 | | | 4 | strncpy (part_rec.partn,itemrq.partnm,5);
149 | | | 5 | strncpy (part_rec.partd," ",25);
150 | | | 6 | db_fdbk = _Rreadk(dbfptr, &part_rec, sizeof(part_rec),
151 | | | 6 | _KEY_EQ, &part_rec.partn, sizeof(part_rec.partn));
152 | | |
153 | | | /*****
154 | | | /* If the part number is found, a positive response to the
155 | | | /* confirmation request is issued, followed by a write
156 | | | /* operation with the requested information. Otherwise,
157 | | | /* a negative response to the confirmation request is issued,
158 | | | /* followed by a write operation with an error message.
159 | | | /*****
160 | | | 7 | if (db_fdbk -> num_bytes > 0) /* if record was found
161 | | |
162 | | | {
163 | | | 8 | _Rformat(icffptr, "ITEMOK ");
164 | | | 9 | _Rwrite(icffptr, 0, 0);
165 | | | 10 | if (check_rc(RC_0000) == NO_MATCH)
166 | | | 11 | cleanup();

```

Figure E-6 (Part 3 of 7). C/400 Inquiry Example – Remote Program

```

167 12 |      strncpy(itemds.partds,part_rec.partd,25);
168 13 |      _Rformat(icffptr, "ITEMDS  ");
169 14 |      _Rwrite(icffptr, &itemds, sizeof(itemds));
170 15 |      if (check_rc(RC_0001) == NO_MATCH)
171 16 |          cleanup();
172   |      }
173   |  else                               /* part description not found. */
174   |  {
175 8 17 |      _Rformat(icffptr, "PGMERR  ");
176 18 |      _Rwrite(icffptr, 0,0);
177 19 |      if (check_rc(RC_0000) == NO_MATCH)
178 20 |          cleanup();
179   |
180 21 |      strncpy(errdes_i_o.errord, part_not_found, 40);
181 22 |      _Rformat(icffptr, "ERRDES  ");
182 23 |      _Rwrite(icffptr, &errdes_i_o, sizeof(errdes_i_o));
183 24 |      if (check_rc(RC_0001) == NO_MATCH)
184 25 |          cleanup();
185   |  }
186   |  }                                     /* end WHILE          */
187   |
188 9 26 | cleanup();
189   | }                                       /* end of main routine  */
190   |
191   | /*****
192   | /*
193   | /* *****
194   | /* *          INTERNAL FUNCTIONS          *
195   | /* *****
196   | /*
197   | /*****
198   |
199   | /*****
200   | /* "OPEN_FILES" function
201   | /*
202   | /* This function opens the database and ICF files, and sets the
203   | /* indicator area for the ICF file.
204   | /*****
205 10 void open_files()
206   | {
207   |
208   | if ((icffptr=_Ropen("T8189ICF","ar+ indicators=y riofb=y"))
209 1 |   == NULL)
210   | {
211 2 |   printf("ICF file failed to open.\n");
212 3 |   exit(ERROR);
213   | }
214   |
6
Line STMT
215 4 | if ((dbfptr=_Ropen("T8189DB", "rr riofb=n")) == NULL)
216   | {
217 5 |   printf("Database file failed to open.\n");
218 6 |   exit(ERROR);
219   | }
220   |
221 11 7 | _Rindara(icffptr, icf_indic);
222   |
223   | }                                       /* end open_files...   */
224   |
225   | /*****
226   | /* "START_CONVERSATION" function
227   | /*
228   | /* This function establishes a conversation with the remote system.
229   | /* The "requesting" program device is acquired.
230   | /*****

```

Figure E-6 (Part 4 of 7). C/400 Inquiry Example – Remote Program

```

231 12 |void start_conversation()
232 |{
233 1 | _Racquire(icffptr, "ICF01 ");
234 2 | if (check_rc(RC_0000) == NO_MATCH)
235 3 | cleanup();
236 |} /* end start_conversation... */
237 |
238 |/*****
239 |/* "GET_CUST_NUM" function */
240 |/* */
241 |/* This subroutine waits for incoming data from the partner */
242 |/* program by issuing the read operation. */
243 |/*****
244 13 |get_cust_num()
245 |{
246 1 | _Rformat(icffptr, "ITEMRQ ");
247 2 | _Rreadn(icffptr, &itemrq, sizeof(itemrq), __DFT);
248 3 | if (check_rc(RC_0014) == MATCH)
249 4 | return(MORE_REQUESTS);
250 | else
251 5 | return(NO_REQUESTS);
252 |} /* end get_cust_num... */
253 |
254 |/*****
255 |/* "CHECK_RC" function */
256 |/* */
257 |/* This function compares the actual ICF return code received */
258 |/* on an operation with an ICF return code that was expected. */
259 |/* If the return codes match, then an indication that the return */
260 |/* codes matched is returned; otherwise, an indication that the */
261 |/* return codes did not match will be returned. */
262 |/*****
263 14 |check_rc(int rc_type)
264 |{
265 1 | comm_fdbk = _Riofbk(icffptr);
266 2 | dsp_icf_fdbk = (XXIOFB_DSP_ICF_T *) ((char *) comm_fdbk +
267 | comm_fdbk->file_dep_fb_offset);
268 |
7
Line STMT *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
269 3 | if (rc_type == RC_0000)
270 | {
271 | if (strcmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&
272 4 | strcmp(dsp_icf_fdbk->minor_ret_code, "00", 2) == 0)
273 5 | return(MATCH);
274 | else
275 6 | return(NO_MATCH);
276 | }
277 | else
278 7 | if (rc_type == RC_0001)
279 | {
280 | if (strcmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&
281 8 | strcmp(dsp_icf_fdbk->minor_ret_code, "01", 2) == 0)
282 9 | return(MATCH);
283 | else
284 10 | return(NO_MATCH);
285 | }
286 | else
287 11 | if (rc_type == RC_0308)
288 | {
289 | if (strcmp(dsp_icf_fdbk->major_ret_code, "03", 2) == 0 &&
290 12 | strcmp(dsp_icf_fdbk->minor_ret_code, "08", 2) == 0)
291 13 | return(MATCH);
292 | else
293 14 | return(NO_MATCH);
294 | }

```

Figure E-6 (Part 5 of 7). C/400 Inquiry Example – Remote Program

```

295 |     else
296 | 15 |         if (rc_type == RC_0014)
297 |     |         {
298 |     |             if (strcmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&
299 | 16 |                 strcmp(dsp_icf_fdbk->minor_ret_code, "14", 2) == 0)
300 | 17 |                 return(MATCH);
301 |     |             else
302 | 18 |                 return(NO_MATCH);
303 |     |             }
304 |     |         else
305 | 19 |             return(NO_MATCH);
306 |     |     }
307 |     |                                     /* end check_rc...          */
308 |     |     /*****
309 |     |     /* "CLEANUP" function.
310 |     |     /*
311 |     |     /* The following code handles the end-of-program processing.
312 |     |     /* This includes the ending of the conversation with
313 |     |     /* the remote system (if conversation is active), and the
314 |     |     /* closing of opened files.
315 |     |     /*****
316 | 15 | void cleanup()
317 |     | {
318 | 16 | 1 | if (check_rc(RC_0308) == MATCH)
319 |     | | {
320 | 2 | |     _Rrelease(icffptr, "ICF01  ");
321 |     | | }
322 |     | | else
8
Line STMT
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
323 | 17 | if ((strcmp(dsp_icf_fdbk->major_ret_code, "80", 2) != 0) &&
324 |     | (strcmp(dsp_icf_fdbk->major_ret_code, "81", 2) != 0) &&
325 | 3 | (strcmp(dsp_icf_fdbk->major_ret_code, "82", 2) != 0))
326 |     | {
327 | 4 |     _Rformat(icffptr, "EOSREC  ");
328 | 5 |     _Rwrite(icffptr, 0, 0);
329 |     | }
330 |
331 | 18 | 6 | _Rclose(icffptr);
332 | 7 | _Rclose(dbfptr);
333 |
334 | 8 | exit(0);
335 |     | }
336 |     |                                     /* end cleanup...          */
                                     * * * * *   E N D   O F   S O U R C E   * * * * *
9
                                     * * * * *   I N C L U D E S   * * * * *
INCNO  Include Name
1      stdio.h
2      stddef.h
3      errno.h
4      signal.h
5      ctype.h
6      stdarg.h
7      stdlib.h
8      string.h
9      stddef.h
10     xxfdbk.h
11     xxasio.h
12     recio.h
13     xxfdbk.h
                                     * * * * *   E N D   O F   I N C L U D E S   * * * * *
Last change      Actual Include Name
90/09/12 14:42:23 QCC/H/STDIO
90/09/12 14:42:22 QCC/H/STDDEF
90/09/12 14:42:18 QCC/H/ERRNO
90/09/12 14:42:22 QCC/H/SIGNAL
90/09/12 14:42:18 QCC/H/CTYPE
90/09/12 14:42:22 QCC/H/STDARG
90/09/12 14:42:24 QCC/H/STDLIB
90/09/12 14:42:24 QCC/H/STRING
90/09/12 14:42:22 QCC/H/STDDEF
90/09/12 14:42:28 QCC/H/XXFDBK
90/09/12 14:42:26 QCC/H/XXASIO
90/09/12 14:42:21 QCC/H/RECIO
90/09/12 14:42:28 QCC/H/XXFDBK

```

Figure E-6 (Part 6 of 7). C/400 Inquiry Example – Remote Program


```

          * * * * * 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 * * * * *
ROUTINE      BLOCK NUMBER SCOPE TYPE
<MAIN>         2          LOCAL MAIN-PROGRAM
__strncmp     149         LOCAL PROCEDURE
__strncpy     151         LOCAL PROCEDURE
QXXACQUIRE   167         LOCAL PROCEDURE
QXXRELEASE   168         LOCAL PROCEDURE
QXXFORMAT    169         LOCAL PROCEDURE
QXXINDARA    170         LOCAL PROCEDURE
__reads      192         LOCAL PROCEDURE
__readk      194         LOCAL PROCEDURE
__rwrite     195         LOCAL PROCEDURE
__rfmt       202         LOCAL PROCEDURE
check_rc     207         ENTRY  PROCEDURE
get_cust_num 208         ENTRY  PROCEDURE
cleanup      209         ENTRY  PROCEDURE
open_files   210         ENTRY  PROCEDURE
start_conversation
              211         ENTRY  PROCEDURE
main         219         ENTRY  PROCEDURE
          * * * * * E N D O F C O M P I L A T I O N * * * * *

```

Figure E-6 (Part 7 of 7). C/400 Inquiry Example – Remote Program

COBOL/400 Local Program for Inquiry Applications (Example 2)

The following explains the structure of the COBOL/400 local program that sends requests to the partner program for processing.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure E-7 on page E-23.

Note: On any type of error that is not expected (for example, an unexpected ICF return code on an I/O operation), the session is ended and the program ends.

1 The files used in the program are described in the file control section.

T8189ICF is the ICF file used to send records to, and receive records from, the partner program. T8189ICF uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

T8189DSP is the name of the display device file that is used to request an entry from the work station and to display the results of the inquiry. T8189DSP uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

2 This section of the program redefines the I/O feedback areas for use within the program.

Refer to the *ICF Programmer's Guide* for a description of the I/O feedback areas.

- 3** The OPEN-FILES, START-CONVERSATION, and GET-CUST-NUM routines are called to open files used by the program, start a conversation with the partner program, and obtain the part number to be queried, respectively.
- 4** The program loops until either F3 is pressed from the work station, which sets the indicator in the indicator area of the display file, or an error occurs in the transaction with the partner program.
- 5** The CLEAN-UP routine is called to perform end-of-program processing.
- 6** The OPEN-FILES routine opens the display and ICF files. A session is implicitly acquired for the work station when T8189DSP is opened.
- 7** The separate indicator area for the display file T8189DSP is initialized.
- 8** The START-CONVERSATION routine establishes a conversation with the partner program.
- 9** The ICF00 program device is explicitly acquired using the ACQUIRE statement. The acquire-program-device operation makes the

program device available for input or output operations.

Note: The program device ICF00 was previously added to the ICF file T8189ICF by the ADDICFDEVE command.

10

An evoke request is issued using a write operation. The write operation is issued using the ICF file record format PGMSTR, which contains the EVOKE, SECURITY, and SYNLVL keywords.

Note: On the EVOKE keyword, the library name is not specified. If the remote system is an AS/400 system, the library list will be used to search for the program. Also, the remote program that is to be started can be any of the remote programs in this appendix or in Appendix F, "Example Programs for Applications Using the CPI Comm Application Program Interface."

11

The HANDLE-INQUIRY routine contains the body of the loop that sends requests to the partner program.

12

The part number is sent to the partner program using a write operation. The write operation is issued using the ICF file record format ITEMRQ, which contains the confirm (CONFIRM keyword) and allow-write (ALWWRT keyword) functions. When these functions are used, the data is flushed, the data flow direction is changed from send to receive, and a confirmation request is sent to the partner program. The partner program must now respond with a positive or negative response.

13

If the partner program responds with a positive response (ICF return code of 0001) to the confirmation request, a read operation is issued using the ICF file record format ITEMDS to receive the part description. However, if

the partner program responds with a negative response (ICF return code of 83C9) to the confirmation request, a read operation is issued using the ICF file record format ERRDES to receive the error message.

14

The get-attributes operation (ACCEPT statement) is issued so that the I/O feedback areas are updated.

15

The GET-CUST-NUM routine is called to display the information returned by the partner program and to obtain the next part number to be queried.

16

The GET-CUST-NUM routine displays the requested information and reads the next number. The part number field will be blank the first time the part number is read.

17

The CLEAN-UP routine performs end-of-program processing.

18

If no error was detected, then a write operation is issued using the ICF file record format PGMEND, which contains the detach (DETACH keyword) function. The release operation (DROP statement) is then issued to detach the program from the session.

19

If an unexpected error was detected and the communications session is still active, a write operation is issued using the ICF file record format EOSREC, which contains the end-of-session (EOS keyword) function. The end-of-session function detaches the program from the session.

Note: If the end-of-session function is issued during an active transaction, APPC will end the session abnormally.

20

The ICF and display files are closed.

```

Program . . . . . : T8189ILS
Library . . . . . : APPCLIB
Source file . . . . . : QATTCBL
Library . . . . . : QUSRTOOL
Source member . . . . . : T8189ILS      09/26/90 08:27:04
Generation severity level . . . . . : 29
Text 'description' . . . . . : *BLANK
Source listing options . . . . . : *NONE
Generation options . . . . . : *NONE
Message limit:
  Number of messages . . . . . : *NOMAX
  Message limit severity . . . . . : 29
Print file . . . . . : QSYSVRT
Library . . . . . : *LIBL
FIPS flagging . . . . . : *NOFIPS *NOSEG *NODEB *NOOBSOLETE
SAA flagging . . . . . : *NOFLAG
Flagging severity . . . . . : 0
Replace program . . . . . : *YES
Target release . . . . . : *CURRENT
User profile . . . . . : *USER
Authority . . . . . : *LIBCRTAUT
Compiler . . . . . : IBM AS/400 COBOL/400
STMT SEQNBR -A 1 B.+...2....3....4....5....6....7..IDENTFCN  S  COPYNAME  CHG DATE
  1 000010 IDENTIFICATION DIVISION.
    000020
  2 000030 PROGRAM-ID. T8189ILS.
    000040
    000050*****
    000060* Program name.....: T8189ILS *
    000070* Program description.: ICF local program *
    000080* Language.....: COBOL/400 *
    000090*
    000100* This program invokes a program to handle part inquiry on *
    000110* the remote system. The acquire operation is used to *
    000120* establish a communications session. A write operation *
    000130* with the evoke function is then issued, which results in *
    000140* the establishment of a conversation with the remote *
    000150* program. A display which prompts the user for the part *
    000160* number for which part information is requested is then *
    000170* displayed. When the user presses Enter, a write operation *
    000180* is issued (the data sent to the partner program is the *
    000190* part number). Note that the write operation was issued *
    000200* with the confirm (CONFIRM) and allow-write (ALWWRT) *
    000210* functions. These functions results in the flushing of the *
    000220* data (to be sent to the partner program), the changing of *
    000230* the data flow direction (the partner program can send the *
    000240* response), and the sending of a confirmation request to *
    000250* the partner program. If the partner program responds *
    000260* with a positive response to the confirmation request (using *
    000270* the RSPCONFIRM function), the ICF return code on the *
    000280* write operation will be set to 0001 (indicating that *
    000290* the part number was found); a read operation is then *
    000300* issued to receive the part description. However, if *
    000310* the partner program responds with a negative response *
    000320* to the confirmation request (using the FAIL function), *
    000330* the ICF return code on the write operation will be set *
    000340* to 83C9 (indicating that the part number was not found); *
    000350* a read operation is issued to receive the error message. *
    000360*
    000370* The error message or part description (depending on *
    000380* whether the part number was found) will be displayed on *
    000390* the screen. *
    000400*
    000410* This program will continue to handle inquiries until the *
    000420* user presses the F3=Exit key. When F3=Exit is pressed, *
    000430* a write operation with the detach (DETACH) function *
    000440* is issued to end the conversation, and program processing *
    000450* ends. *
    000460*

```

Figure E-7 (Part 1 of 7). COBOL/400 Inquiry Example – Local Program

```

000470* NOTE 1: If an unexpected ICF return code is received on *
000480* any of the read or write operations, the *
000490* program will abnormally end the conversation (if *
000500* it is still active), and program processing will *
000510* end. *
000520* *
000530* NOTE 2: On the receive operation, if the actual received *
000540* data length (obtained from the I/O feedback area) *
000550* does not match what was expected, or if the *
STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME CHG DATE
000560* ICF return code is not 0000 (indication that *
000570* the partner program is ready to receive data), the *
000580* program will abnormally end the conversation (if *
000590* it is still active), and program processing will *
000600* end. *
000610* *
000620* NOTE 3: This program can start ANY of the "remote" *
000630* program examples in the APPC Programmer's *
000640* Guide by changing the PGMID variable to the *
000650* remote program that is to be started. *
000660* *
000670*****
000680
3 000690 ENVIRONMENT DIVISION.
000700
4 000710 CONFIGURATION SECTION.
000720
5 000730 SOURCE-COMPUTER. IBM-AS400.
6 000740 OBJECT-COMPUTER. IBM-AS400.
7 000750 SPECIAL-NAMES. I-O-FEEDBACK IS IO-FEEDBACK
8 000760 OPEN-FEEDBACK IS OPEN-FBA.
000770
9 000780 INPUT-OUTPUT SECTION.
000790
10 000800 FILE-CONTROL.
000810
1 11 000820 SELECT T8189ICF ASSIGN TO WORKSTATION-T8189ICF
12 000830 ORGANIZATION IS TRANSACTION
13 000840 CONTROL-AREA IS TR-CTL-AREA
14 000850 FILE STATUS IS STATUS-IND MAJMIN.
15 000860 SELECT T8189DSP ASSIGN TO WORKSTATION-T8189DSP
16 000870 ORGANIZATION IS TRANSACTION
17 000880 CONTROL-AREA IS DISPLAY-FEEDBACK
18 000890 FILE STATUS IS STATUS-DSP.
000900
19 000910 DATA DIVISION.
000920
20 000930 FILE SECTION.
000940
000950*****
000960* File description for the ICF file. *
000970*****
000980
21 000990 FD T8189ICF
22 001000 LABEL RECORDS ARE STANDARD.
23 001010 01 APPCREC.
24 001020 COPY DDS-ALL-FORMATS-I-O OF T8189ICF.
25 +000001 05 T8189ICF-RECORD PIC X(40).
+000002* INPUT FORMAT:PGMSTR FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000003* <-ALL-FMTS
+000004* 05 PGMSTR-I REDEFINES T8189ICF-RECORD. <-ALL-FMTS
+000005* OUTPUT FORMAT:PGMSTR FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000006* <-ALL-FMTS
26 +000007 05 PGMSTR-O REDEFINES T8189ICF-RECORD. <-ALL-FMTS
27 +000008 06 PGMID PIC X(10). <-ALL-FMTS

```

Figure E-7 (Part 2 of 7). COBOL/400 Inquiry Example — Local Program

```

STMT SEQNBR -A 1 B.....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME CHG DATE
+000009* I-0 FORMAT:ITEMRQ FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000010* <-ALL-FMTS
28 +000011 05 ITEMRQ REDEFINES T8189ICF-RECORD. <-ALL-FMTS
29 +000012 06 PARTNM PIC X(5). <-ALL-FMTS
+000013* I-0 FORMAT:ITEMDS FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000014* <-ALL-FMTS
30 +000015 05 ITEMDS REDEFINES T8189ICF-RECORD. <-ALL-FMTS
31 +000016 06 PARTDS PIC X(25). <-ALL-FMTS
+000017* I-0 FORMAT:ERRDES FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000018* <-ALL-FMTS
32 +000019 05 ERRDES REDEFINES T8189ICF-RECORD. <-ALL-FMTS
33 +000020 06 ERRORR PIC X(40). <-ALL-FMTS
+000021* I-0 FORMAT:PGMEND FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000022* <-ALL-FMTS
+000023* 05 PGMEND REDEFINES T8189ICF-RECORD. <-ALL-FMTS
+000024* I-0 FORMAT:EOSREC FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000025* <-ALL-FMTS
+000026* 05 EOSREC REDEFINES T8189ICF-RECORD. <-ALL-FMTS
+000027* I-0 FORMAT:PGMERR FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000028* <-ALL-FMTS
+000029* 05 PGMERR REDEFINES T8189ICF-RECORD. <-ALL-FMTS
+000030* I-0 FORMAT:ITEMOK FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000031* <-ALL-FMTS
+000032* 05 ITEMOK REDEFINES T8189ICF-RECORD. <-ALL-FMTS
001030
001040*****
001050* File description for the display file. *
001060*****
001070
34 001080 FD T8189DSP
35 001090 LABEL RECORDS ARE STANDARD.
36 001100 01 DSPREC.
37 001110 COPY DDS-ALL-FORMATS-I-0 OF T8189DSP.
38 +000001 05 T8189DSP-RECORD PIC X(70). <-ALL-FMTS
+000002* INPUT FORMAT:PROMPT FROM FILE T8189DSP OF LIBRARY APPCLIB <-ALL-FMTS
+000003* <-ALL-FMTS
39 +000004 05 PROMPT-I REDEFINES T8189DSP-RECORD. <-ALL-FMTS
40 +000005 06 PARTN PIC X(5). <-ALL-FMTS
+000006* OUTPUT FORMAT:PROMPT FROM FILE T8189DSP OF LIBRARY APPCLIB <-ALL-FMTS
+000007* <-ALL-FMTS
41 +000008 05 PROMPT-O REDEFINES T8189DSP-RECORD. <-ALL-FMTS
42 +000009 06 PARTN PIC X(5). <-ALL-FMTS
43 +000010 06 PARTD PIC X(25). <-ALL-FMTS
44 +000011 06 ERRORL PIC X(40). <-ALL-FMTS
001120
45 001130 WORKING-STORAGE SECTION.
001140
46 001150 77 STATUS-IND PIC XX.
47 001160 77 STATUS-DSP PIC XX.
001170
48 001180 01 TR-CTL-AREA.
49 001190 05 FILLER PIC X(2).
50 001200 05 PGM-DEV-NME PIC X(10).
51 001210 05 RCD-FMT-NME PIC X(10).
001220
STMT SEQNBR -A 1 B.....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME CHG DATE
52 001230 01 DSPF-INDIC-AREA.
53 001240 05 CMD3 PIC 1 INDIC 99.
54 001250 88 CMD3-ON VALUE B"1".
55 001260 88 CMD3-OFF VALUE B"0".
001270

```

Figure E-7 (Part 3 of 7). COBOL/400 Inquiry Example – Local Program

```

2 56 001280 01 IO-FBA.
57 001290 05 COMMON-IO-FBA.
58 001300 10 FILLER PIC X(144).
59 001310 05 FILE-DEP-IO-FBA.
60 001320 10 FILLER PIC X(5).
61 001330 10 ACTUAL-LENGTH PIC 9(9) COMP-4.
62 001340 10 FILLER PIC X(25).
63 001350 10 MAJ-MIN-S.
64 001360 15 MAJ-S PIC X(2).
65 001370 15 MIN-S PIC X(2).
66 001380 10 ERDTA PIC X(8).
67 001390 05 FILLER PIC X(21).
001400
68 001410 01 MAJMIN.
69 001420 05 MAJCOD PIC XX.
70 001430 05 MINCOD PIC XX.
001440
71 001450 01 DISPLAY-FEEDBACK.
72 001460 05 CMD-KEY PIC XX.
73 001470 05 FILLER PIC X(10).
74 001480 05 RCD-FMT PIC X(10).
001490
75 001500 01 REQUEST-LENGTH PIC 9(9) COMP-4.
001510
76 001520 01 ERR-SWITCH PIC 9(9) COMP-4.
77 001530 88 NO-ERROR-OCCURRED VALUE 0.
78 001540 88 ERROR-OCCURRED VALUE 1.
001550
79 001560 PROCEDURE DIVISION.
001570
001580 DECLARATIVES.
001590 END DECLARATIVES.
001600
001610*****
001620* START OF PROGRAM *
001630* *
001640* Files are opened, a conversation with the *
001650* remote program is started, and the part inquiry *
001660* screen is displayed. Inquiries are handled until *
001670* the user presses the F3=Exit key, in which case *
001680* the conversation will be ended and the program will end. *
001690*****
001700
001710 START-PROGRAM SECTION.
001720
001730 START-PROGRAM-PARAGRAPH.
001740
3 80 001750 PERFORM OPEN-FILES.
81 001760 PERFORM START-CONVERSATION.
001770
STMT SEQNBR -A 1 B..+...2....+...3....+...4....+...5....+...6....+...7..IDENTFCN S COPYNAME CHG DATE
82 001780 PERFORM GET-CUST-NUM.
001790
4 83 001800 PERFORM HANDLE-INQUIRY UNTIL CMD3-ON.
001810
84 001820 SET NO-ERROR-OCCURRED TO TRUE.
5 85 001830 PERFORM CLEAN-UP.
001840
001850
001860*****
001870* "OPEN-FILES" routine. *
001880* *
001890* This routine opens the display and ICF files. *
001900*****
001910
6 001920 OPEN-FILES.
001930
86 001940 OPEN I-0 T8189ICF T8189DSP.
87 001950 MOVE SPACES TO DSPREC.
7 88 001960 MOVE ZEROS TO DSPF-INDIC-AREA.

```

Figure E-7 (Part 4 of 7). COBOL/400 Inquiry Example – Local Program

```

001970
001980*****
001990* "START-CONVERSATION" routine. *
002000* *
002010* This subroutine establishes a conversation with the *
002020* remote program. *
002030*****
002040
8 002050 START-CONVERSATION.
002060
89 002070 SET ERROR-OCCURRED TO TRUE.
002080
002090*****
002100* The acquire operation is issued. *
002110*****
9 90 002120 ACQUIRE "ICF00 " FOR T8189ICF.
91 002130 MOVE "ICF00 " TO PGM-DEV-NME.
002140
92 002150 IF MAJMIN = "0000" THEN
002160 NEXT SENTENCE
002170 ELSE
93 002180 PERFORM CLEAN-UP.
002190
002200*****
002210* A write operation with the evoke function is issued so *
002220* that a conversation can be started. *
002230*****
94 002240 MOVE "T8189ILT" TO PGMID.
10 95 002250 WRITE APPCREC FORMAT IS "PGMSTR"
002260 TERMINAL IS PGM-DEV-NME.
002270
96 002280 IF MAJMIN = "0000" THEN
002290 NEXT SENTENCE
002300 ELSE
97 002310 PERFORM CLEAN-UP.
002320
STMT SEQNBR -A 1 B.+. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . IDENTFCN S COPYNAME CHG DATE
002330
002340*****
002350* "HANDLE-INQUIRY" routine. *
002360* *
002370* This is the main loop of the program. Process the part *
002380* number keyed in by the user until F3 (CMD3) is pressed. *
002390*****
002400
11 002410 HANDLE-INQUIRY.
002420
002430*****
002440* The part number that the user has requested information *
002450* for is sent to the remote program using the write *
002460* operation with the confirm and allow-write functions. *
002470*****
98 002480 MOVE PARTN OF PROMPT-I TO PARTNM OF ITEMRQ.
12 99 002490 WRITE APPCREC FORMAT IS "ITEMRQ"
002500 TERMINAL IS PGM-DEV-NME.
002510
002520*****
002530* The read operation is issued to receive the response *
002540* from the remote program (the response can either be *
002550* an error message or the part description, depending *
002560* on whether the part was found or not). *
002570*****

```

Figure E-7 (Part 5 of 7). COBOL/400 Inquiry Example – Local Program

```

13 100 002580   IF MAJMIN = "0001" THEN
101 002590       MOVE 25 TO REQUEST-LENGTH
102 002600       MOVE SPACES TO ERRORL
103 002610       READ T8189ICF FORMAT IS "ITEMDS"
002620           TERMINAL IS PGM-DEV-NME
104 002630       MOVE PARTDS OF ITEMDS TO PARTD
002640   ELSE
105 002650       IF MAJMIN = "83C9" THEN
106 002660           MOVE 40 TO REQUEST-LENGTH
107 002670           MOVE SPACES TO PARTD
108 002680           READ T8189ICF FORMAT IS "ERRDES"
109 002690           MOVE ERRORD OF ERRDES TO ERRORL
002700   ELSE
110 002710       PERFORM CLEAN-UP.
002720
14 111 002730   ACCEPT IO-FBA FROM IO-FEEDBACK.
002740
112 002750   IF MAJMIN = "0000" AND
002760       ACTUAL-LENGTH = REQUEST-LENGTH THEN
002770       NEXT SENTENCE
002780   ELSE
113 002790       PERFORM CLEAN-UP.
002800
15 114 002810   PERFORM GET-CUST-NUM.
002820
002830*****
002840* "GET-CUST-NUM" routine. *
002850* *
002860* Get a customer number from the display. *
002870*****
STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME CHG DATE
002880
16 002890 GET-CUST-NUM.
002900
115 002910   WRITE DSPREC FORMAT IS "PROMPT".
116 002920   READ T8189DSP INDICATORS ARE DSPF-INDIC-AREA.
002930
002940*****
002950* "CLEAN-UP" routine. *
002960* *
002970* The following code handles the end-of-program processing. *
002980* This includes the ending of the conversation with *
002990* the remote system (if conversation is active) by either *
003000* issuing a write operation with the detach function *
003010* followed by a release operation (for non-error conditions),*
003020* or by issuing a write operation with the end-of-session *
003030* function (for error conditions). *
003040*****
003050
003060 CLEAN-UP.
003070
117 003080   IF MAJCOD = "80" OR
003090       MAJCOD = "81" OR
003100       MAJCOD = "82" THEN
003110       NEXT SENTENCE
003120   ELSE
18 118 003130       IF NO-ERROR-OCCURRED THEN
119 003140           WRITE APPREC FORMAT IS "PGMEND"
120 003150           DROP "ICF00 " FROM T8189ICF
003160       ELSE
19 121 003170           WRITE APPREC FORMAT IS "EOSREC".
003180
20 122 003190   CLOSE T8189DSP T8189ICF.
003200
123 003210   STOP RUN.
          * * * * *   E N D   O F   S O U R C E   * * * * *

```

Figure E-7 (Part 6 of 7). COBOL/400 Inquiry Example — Local Program


```

STMT
*      MSGID: LBL0904 SEVERITY: 00 SEQNBR:
      Message . . . . : Unexpected source member type.
* 24  MSGID: LBL0600 SEVERITY: 10 SEQNBR: 001020
      Message . . . . : No INPUT fields found for format PGMSTR.
* 24  MSGID: LBL0600 SEVERITY: 10 SEQNBR: 001020
      Message . . . . : No INPUT fields found for format PGMEND.
* 24  MSGID: LBL0600 SEVERITY: 10 SEQNBR: 001020
      Message . . . . : No INPUT fields found for format EOSREC.
* 24  MSGID: LBL0600 SEVERITY: 10 SEQNBR: 001020
      Message . . . . : No INPUT fields found for format PGMERR.
* 24  MSGID: LBL0600 SEVERITY: 10 SEQNBR: 001020
      Message . . . . : No INPUT fields found for format ITEMOK.
      * * * * * E N D   O F   M E S S A G E S   * * * * *

                                Message Summary
Total      Info(0-4)      Warning(5-19)      Error(20-29)      Severe(30-39)      Terminal(40-99)
   6             1             5             0             0             0
Source records read . . . . . : 321
Copy records read . . . . . : 43
Copy members processed . . . . . : 2
Sequence errors . . . . . : 0
Highest severity message issued . . : 10
LBL0901 00 Program T8189ILS created in library APPCLIB.
      * * * * * E N D   O F   C O M P I L A T I O N   * * * * *

```

Figure E-7 (Part 7 of 7). COBOL/400 Inquiry Example – Local Program

COBOL/400 Remote Program for Inquiry Application (Example 2)

The following explains the structure of the COBOL/400 remote program that handles requests sent by the partner program.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure E-8 on page E-31.

Note: On any type of error that is not expected (for example, an unexpected ICF return code on an I/O operation), the session is ended and the program ends.

- 1** The file division section defines the files used in the program.
T8189ICF is the ICF file used to receive records from, and send records to, the partner program. T8189ICF uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.
T8189DB is the database file that contains the valid part numbers and part descriptions.
- 2** This section defines the I/O feedback areas for use within the program.
- 3** The OPEN-FILES, START-CONVERSATION, and GET-CUST-NUM routines are called to open files used by the program, start a conversation

with the partner program, and wait on a request by the partner program, respectively.

- 4** The program loops until there are no more requests to process, or an error occurs in the transaction with the partner program.
- 5** The CLEAN-UP routine is called to perform end-of-program processing.
- 6** The OPEN-FILES routine opens the database and ICF files.
- 7** The START-CONVERSATION routine establishes a conversation with the partner program by explicitly acquiring the ICF01 program device using the ACQUIRE statement.
Note: The program device ICF01 was previously added to the ICF file T8189ICF by the ADDICFDEVE command.
- 8** The HANDLE-INQUIRY routine contains the body of the loop that handles requests from the partner program.
- 9** A search of the database file is performed using the part number received from the partner program as the key.
- 10** If the part number is found in the database file, a write operation is issued using the ICF

file record format ITEMOK, which contains the respond-to-confirm (RSPCONFIRM keyword) function. As a result, a positive response to the received confirmation request is sent to the partner program. This is followed by a second write operation using the ICF file record format ITEMDS, which contains the requested information to be sent and the allow-write (ALWWRT keyword) function. Using the allow-write function results in the flushing of the data and the changing of the data flow direction.

11 If the part number is not found in the database file, a write operation is issued using the ICF file record format PGMERR, which contains the fail (FAIL keyword) function. As a result, a negative response to the received confirmation request is sent to the partner program. This is followed by a second write operation using the ICF file record format ERRDES, which contains the error message to be sent and the allow-write function.

12 If the session with the partner program is still active, the GET-CUST-NUM routine is called to wait on a request by the partner program.

13 The GET-CUST-NUM routine waits for a request from the partner program by issuing a read

operation using the ICF file record format ITEMRQ.

Note: A transaction is processed if data is received with a turnaround indication, and the partner program requested confirmation. This is indicated by the ICF return code 0014.

14 The CLEAN-UP routine performs end-of-program processing.

15 If no error was detected, a release operation (DROP statement) is issued to detach the program from the session.

16 If an unexpected error was detected and the communications session is still active, a write operation is issued using the ICF file record format EOSREC, which contains the end-of-session (EOS keyword) function. The end-of-session function detaches the program from the session.

Note: If the end-of-session function is issued during an active transaction, APFC will end the session abnormally.

17 The ICF and database files are closed.

```

Program . . . . . : T8189ILT
Library . . . . . : APPCLIB
Source file . . . . . : QATTCBL
Library . . . . . : QUSRTOOL
Source member . . . . . : T8189ILT 09/26/90 08:27:06
Generation severity level . . . . . : 29
Text 'description' . . . . . : *BLANK
Source listing options . . . . . : *NONE
Generation options . . . . . : *NONE
Message limit:
Number of messages . . . . . : *NOMAX
Message limit severity . . . . . : 29
Print file . . . . . : QSYSVRT
Library . . . . . : *LIBL
FIPS flagging . . . . . : *NOFIPS *NOSEG *NODEB *NOOBSOLETE
SAA flagging . . . . . : *NOFLAG
Flagging severity . . . . . : 0
Replace program . . . . . : *YES
Target release . . . . . : *CURRENT
User profile . . . . . : *USER
Authority . . . . . : *LIBCRTAUT
Compiler . . . . . : IBM AS/400 COBOL/400
STMT SEQNBR -A 1 B.+...2....3....+...4....+...5....+...6....+...7..IDENTFCN S COPYNAME CHG DATE
1 000010 IDENTIFICATION DIVISION.
000020
2 000030 PROGRAM-ID. T8189ILT.
000040
000050*****
000060* Program name.....: T8189ILT *
000070* Program description.: ICF remote program *
000080* Language.....: COBOL/400 *
000090* *
000100* This program accepts the incoming conversation by issuing *
000110* the acquire operation to acquire the requesting program *
000120* device. A read operation is then issued to receive the *
000130* part number from the remote system. The read operation *
000140* completes with an ICF return code of 0014, indicating *
000150* that data with a turnaround indication was received, and *
000160* that the partner program also requested confirmation. The *
000170* database file T8189DB is searched for the received part *
000180* number. If the part number is found, a write operation *
000190* with the respond-to-confirm (RSPCONFIRM) function is *
000200* issued, followed by a write operation containing the *
000210* part description corresponding to the part number *
000220* retrieved from the database file. However, if the part *
000230* number is not found, a write operation with the *
000240* negative-response (FAIL) function is issued, followed by *
000250* a write operation containing an error message describing *
000260* the error. The write operation sending either the part *
000270* description or the error message is issued with the *
000280* allow-write (ALWVRT) function. Using the allow-write *
000290* function results in the flushing of the data and the *
000300* changing of the data flow direction. The partner program *
000310* can send more inquiries. *
000320* *
000330* This program will continue to handle inquiries from the *
000340* partner program until a detach indication is received. *
000350* Then the program ends. *
000360* *
000370* NOTE 1: If an unexpected ICF return code is received on *
000380* any of the read or write operations, the *
000390* program will abnormally end the conversation (if *
000400* it is still active), and program processing will *
000410* end. *
000420* *
000430* NOTE 2: On the receive operation, if the actual received *
000440* data length (obtained from the I/O feedback area) *
000450* does not match what was expected, or if the *
000460* ICF return code is not 0014 (indication that *

```

Figure E-8 (Part 1 of 6). COBOL/400 Inquiry Example – Remote Program

```

000470*      data was received with a turnaround indicator,      *
000480*      and partner program requested confirmation), the      *
000490*      program will abnormally end the conversation (if      *
000500*      it is still active), and program processing will      *
000510*      end.                                                    *
000520*      *                                                        *
000530* NOTE 3: This program can be started by ANY of the      *
000540*      "local" program examples in the APPC Programmer's      *
000550*      Guide.                                                  *
STMT SEQNBR -A 1 B..+...2...+...3...+...4...+...5...+...6...+...7..IDENTFCN S COPYNAME  CHG DATE
000560*      *
000570*****
000580
3 000590 ENVIRONMENT DIVISION.
000600
4 000610 CONFIGURATION SECTION.
000620
5 000630 SOURCE-COMPUTER. IBM-AS400.
6 000640 OBJECT-COMPUTER. IBM-AS400.
7 000650 SPECIAL-NAMES. I-O-FEEDBACK IS IO-FEEDBACK
8 000660 OPEN-FEEDBACK IS OPEN-FBA.
000670
9 000680 INPUT-OUTPUT SECTION.
000690
10 000700 FILE-CONTROL.
000710
1 11 000720 SELECT T8189ICF ASSIGN TO WORKSTATION-T8189ICF
12 000730 ORGANIZATION IS TRANSACTION
13 000740 CONTROL-AREA IS TR-CTL-AREA
14 000750 FILE STATUS IS STATUS-IND MAJMIN.
15 000760 SELECT T8189DB ASSIGN TO DATABASE-T8189DB
16 000770 ORGANIZATION IS INDEXED
17 000780 ACCESS IS RANDOM
18 000790 RECORD KEY IS ITEMNM.
000800
19 000810 DATA DIVISION.
000820
20 000830 FILE SECTION.
000840
000850*****
000860* File description for the ICF file. *
000870*****
000880
21 000890 FD T8189ICF
22 000900 LABEL RECORDS ARE STANDARD.
23 000910 01 APPCREC.
24 000920 COPY DDS-ALL-FORMATS-I-O OF T8189ICF.
25 +000001 05 T8189ICF-RECORD PIC X(40). <-ALL-FMTS
+000002* INPUT FORMAT:PGMSTR FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000003* <-ALL-FMTS
+000004* 05 PGMSTR-I REDEFINES T8189ICF-RECORD. <-ALL-FMTS
+000005* OUTPUT FORMAT:PGMSTR FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000006* <-ALL-FMTS
26 +000007 05 PGMSTR-0 REDEFINES T8189ICF-RECORD. <-ALL-FMTS
27 +000008 06 PGMID PIC X(10). <-ALL-FMTS
+000009* I-O FORMAT:ITEMRQ FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000010* <-ALL-FMTS
28 +000011 05 ITEMRQ REDEFINES T8189ICF-RECORD. <-ALL-FMTS
29 +000012 06 PARTNM PIC X(5). <-ALL-FMTS
+000013* I-O FORMAT:ITEMDS FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000014* <-ALL-FMTS
30 +000015 05 ITEMDS REDEFINES T8189ICF-RECORD. <-ALL-FMTS
31 +000016 06 PARTDS PIC X(25). <-ALL-FMTS
+000017* I-O FORMAT:ERRDES FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000018* <-ALL-FMTS
STMT SEQNBR -A 1 B..+...2...+...3...+...4...+...5...+...6...+...7..IDENTFCN S COPYNAME  CHG DATE
32 +000019 05 ERRDES REDEFINES T8189ICF-RECORD. <-ALL-FMTS
33 +000020 06 ERROR PIC X(40). <-ALL-FMTS
+000021* I-O FORMAT:PGMEND FROM FILE T8189ICF OF LIBRARY APPCLIB <-ALL-FMTS
+000022* <-ALL-FMTS

```

Figure E-8 (Part 2 of 6). COBOL/400 Inquiry Example — Remote Program

```

+000023*      05 PGMEND      REDEFINES T8189ICF-RECORD.          <-ALL-FMTS
+000024*      I-O FORMAT:EOSREC      FROM FILE T8189ICF  OF LIBRARY APPCLIB  <-ALL-FMTS
+000025*                                          <-ALL-FMTS
+000026*      05 EOSREC      REDEFINES T8189ICF-RECORD.          <-ALL-FMTS
+000027*      I-O FORMAT:PGMERR      FROM FILE T8189ICF  OF LIBRARY APPCLIB  <-ALL-FMTS
+000028*                                          <-ALL-FMTS
+000029*      05 PGMERR      REDEFINES T8189ICF-RECORD.          <-ALL-FMTS
+000030*      I-O FORMAT:ITEMOK      FROM FILE T8189ICF  OF LIBRARY APPCLIB  <-ALL-FMTS
+000031*                                          <-ALL-FMTS
+000032*      05 ITEMOK      REDEFINES T8189ICF-RECORD.          <-ALL-FMTS
000930
000940*****
000950* File description for the database file.          *
000960*****
000970
34 000980 FD T8189DB
35 000990 LABEL RECORDS ARE STANDARD.
36 001000 01 DBREC.
37 001010 COPY DDS-ALL-FORMATS OF T8189DB.
38 +000001      05 T8189DB-RECORD PIC X(30).          <-ALL-FMTS
+000002*      I-O FORMAT:DBRCD      FROM FILE T8189DB    OF LIBRARY APPCLIB  <-ALL-FMTS
+000003*                                          <-ALL-FMTS
+000004*      USER SUPPLIED KEY BY RECORD KEY CLAUSE          <-ALL-FMTS
39 +000005      05 DBRCD      REDEFINES T8189DB-RECORD.          <-ALL-FMTS
40 +000006      06 ITEMNH      PIC X(5).          <-ALL-FMTS
41 +000007      06 ITEM D      PIC X(25).          <-ALL-FMTS
001020
42 001030 WORKING-STORAGE SECTION.
001040
43 001050 77 STATUS-IND      PIC XX.
001060
44 001070 01 TR-CTL-AREA.
45 001080      05 FILLER      PIC X(2).
46 001090      05 PGM-DEV-NME  PIC X(10).
47 001100      05 RCD-FMT-NME  PIC X(10).
001110
2 48 001120 01 IO-FBA.
49 001130      05 COMMON-IO-FBA.
50 001140      10 FILLER      PIC X(144).
51 001150      05 FILE-DEP-IO-FBA.
52 001160      10 FILLER      PIC X(5).
53 001170      10 ACTUAL-LENGTH PIC 9(9) COMP-4.
54 001180      10 FILLER      PIC X(25).
55 001190      10 MAJMIN-S.
56 001200      15 MAJ-S      PIC X(2).
57 001210      15 MIN-S      PIC X(2).
58 001220      10 ERDTA      PIC X(8).
59 001230      05 FILLER      PIC X(21).
001240
60 001250 01 ERROR-FND      PIC X.
61 001260 01 MAJMIN.
STMT SEQNBR -A 1 B..+...2....+...3....+...4....+...5....+...6....+...7..IDENTFCN S COPYNAME CHG DATE
62 001270      05 MAJCOD      PIC XX.
63 001280      05 MINCOD      PIC XX.
001290
64 001300 01 NOT-FND-MSG      PIC X(40)
65 001310      VALUE "The requested part was not found.  ".
001320
66 001330 PROCEDURE DIVISION.
001340
001350 DECLARATIVES.
001360 END DECLARATIVES.
001370
001380 START-PROGRAM SECTION.
001390
001400 START-PROGRAM-PARAGRAPH.
001410

```

Figure E-8 (Part 3 of 6). COBOL/400 Inquiry Example – Remote Program

```

001420*****
001430* START OF PROGRAM *
001440* *
001450* Files are opened, a conversation with the *
001460* remote program is started, and the part inquiry *
001470* processing starts. Inquiries are handled until a *
001480* detach indication is received. *
001490*****
001500
3 67 001510 PERFORM OPEN-FILES.
68 001520 PERFORM START-CONVERSATION.
69 001530 PERFORM GET-CUST-NUM.
001540
4 70 001550 PERFORM HANDLE-INQUIRY UNTIL
001560 NOT (MAJMIN IS EQUAL TO "0014").
001570
5 71 001580 PERFORM CLEAN-UP.
001590
001600*****
001610* "OPEN-FILES" routine. *
001620* *
001630* This routine opens the ICF and database files. *
001640*****
001650
6 001660 OPEN-FILES.
001670
72 001680 OPEN I-O T8189ICF T8189DB.
001690
001700*****
001710* "START-CONVERSATION" routine. *
001720* *
001730* This routine establishes a conversation with the *
001740* remote program. *
001750*****
001760
7 001770 START-CONVERSATION.
001780
001790*****
001800* The acquire operation is issued. *
001810*****
STMT SEQNBR -A 1 B..+...2...+...3...+...4...+...5...+...6...+...7..IDENTFCN S COPYNAME CHG DATE
73 001820 MOVE "ICF01 " TO PGM-DEV-NME.
74 001830 ACQUIRE "ICF01 " FOR T8189ICF.
001840
75 001850 IF MAJMIN = "0000" THEN
001860 NEXT SENTENCE
001870 ELSE
76 001880 PERFORM CLEAN-UP.
001890
001900*****
001910* "HANDLE-INQUIRY" routine. *
001920* *
001930* This is the main loop of the program. Process inquiry *
001940* request until conversation is ended. *
001950*****
001960
8 001970 HANDLE-INQUIRY.
77 001980 MOVE "0" TO ERROR-FND.
001990
002000*****
002010* A search of the database file is done using the part *
002020* number as the key. *
002030*****
9 78 002040 READ T8189DB FORMAT IS "DBRCD"
79 002050 INVALID KEY MOVE "1" TO ERROR-FND.
002060

```

Figure E-8 (Part 4 of 6). COBOL/400 Inquiry Example – Remote Program

```

002070******
002080* If the part number is found, a positive response to the *
002090* confirmation request is issued, followed by a write *
002100* operation with the requested information. Otherwise, *
002110* a negative response to the confirmation request is issued, *
002120* followed by a write operation with an error message. *
002130******
10 80 002140     IF ERROR-FND = "0" THEN
81 002150     WRITE APPREC FORMAT IS "ITEMOK"
002160     TERMINAL IS PGM-DEV-NME
82 002170     IF MAJMIN = "0000" THEN
002180     NEXT SENTENCE
002190     ELSE
83 002200     PERFORM CLEAN-UP
002210     END-IF
84 002220     MOVE ITEMID TO PARTDS OF ITEMDS
85 002230     WRITE APPREC FORMAT IS "ITEMDS"
002240     TERMINAL IS PGM-DEV-NME
002250     ELSE
11 86 002260     MOVE NOT-FND-MSG TO ERROR OF ERRDES
87 002270     WRITE APPREC FORMAT IS "PGHERR"
002280     TERMINAL IS PGM-DEV-NME
88 002290     IF MAJMIN = "0000" THEN
002300     NEXT SENTENCE
002310     ELSE
89 002320     PERFORM CLEAN-UP
002330     END-IF
90 002340     WRITE APPREC FORMAT IS "ERRDES"
002350     TERMINAL IS PGM-DEV-NME.
002360
STMT SEQNBR -A 1 B.+. . . . 2. . . . 3. . . . 4. . . . 5. . . . 6. . . . 7. . IDENTFCN S COPYNAME CHG DATE
12 91 002370     IF MAJMIN = "0001" THEN
92 002380     PERFORM GET-CUST-NUM
002390     ELSE
93 002400     PERFORM CLEAN-UP.
002410
002420
002430******
002440* "GET-CUST-NUM" routine. *
002450* *
002460* This subroutine waits for incoming data from the partner *
002470* program by issuing the read operation. *
002480******
002490
13 002500 GET-CUST-NUM.
94 002510     READ T8189ICF FORMAT IS "ITEMRQ".
95 002520     ACCEPT IO-FBA FROM IO-FEEDBACK.
002530
96 002540     IF MAJMIN = "0014" THEN
97 002550     IF ACTUAL-LENGTH = 5 THEN
002560     NEXT SENTENCE
002570     ELSE
98 002580     PERFORM CLEAN-UP
002590     ELSE
002600     NEXT SENTENCE.
002610
002620******
002630* "CLEAN-UP" routine. *
002640* *
002650* The following code handles the end-of-program processing. *
002660* This includes the ending of the conversation with *
002670* the remote system (if conversation is active), and the *
002680* closing of opened files. *
002690******

```

Figure E-8 (Part 5 of 6). COBOL/400 Inquiry Example – Remote Program

```

002700
14 002710 CLEAN-UP.
002720
002730
99 002740     IF MAJCOD = "80" OR
002750     MAJCOD = "81" OR
002760     MAJCOD = "82" THEN
002770     NEXT SENTENCE
002780     ELSE
15 100 002790     IF MAJMIN = "0308" THEN
101 002800     DROP "ICF01 " FROM T8189ICF
002810     ELSE
16 102 002820     WRITE APPCREC FORMAT IS "EOSREC"
002830     TERMINAL IS PGM-DEV-NME.
002840
002850
17 103 002860     CLOSE T8189DB T8189ICF.
002870
104 002880     STOP RUN.
002890

                * * * * *   E N D   O F   S O U R C E   * * * * *

STMT
*   MSGID: LBL0904 SEVERITY: 00 SEQNBR:
Message . . . . : Unexpected source member type.
* 24 MSGID: LBL0600 SEVERITY: 10 SEQNBR: 000920
Message . . . . : No INPUT fields found for format PGMSTR.
* 24 MSGID: LBL0600 SEVERITY: 10 SEQNBR: 000920
Message . . . . : No INPUT fields found for format PGMEND.
* 24 MSGID: LBL0600 SEVERITY: 10 SEQNBR: 000920
Message . . . . : No INPUT fields found for format EOSREC.
* 24 MSGID: LBL0600 SEVERITY: 10 SEQNBR: 000920
Message . . . . : No INPUT fields found for format PGMERR.
* 24 MSGID: LBL0600 SEVERITY: 10 SEQNBR: 000920
Message . . . . : No INPUT fields found for format ITEMOK.
                * * * * *   E N D   O F   M E S S A G E S   * * * * *

                          Message Summary
Total      Info(0-4)    Warning(5-19)    Error(20-29)    Severe(30-39)    Terminal(40-99)
   6         1           5             0              0                0
Source records read . . . . . : 289
Copy records read . . . . . : 39
Copy members processed . . . . . : 2
Sequence errors . . . . . : 0
Highest severity message issued . . : 10
LBL0901 00 Program T8189ILT created in library APPCLIB.
                * * * * *   E N D   O F   C O M P I L A T I O N   * * * * *

```

Figure E-8 (Part 6 of 6). COBOL/400 Inquiry Example — Remote Program

RPG/400 Local Program for Inquiry Applications (Example 3)

The following explains the structure of the RPG/400 local program that sends requests to the partner program for processing.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure E-9 on page E-39.

Note: On any type of error that is not expected (for example, an unexpected ICF return code on an I/O operation), the session is ended and the program ends.

1

The files used in the program are described in the file specifications section.

T8189ICF is the ICF file used to send records to, and receive records from, the partner program. T8189ICF uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

T8189DSP is the name of the display device file that is used to request an entry from the work station and to display the results of the inquiry. T8189DSP uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

All files are implicitly opened at the beginning of the RPG/400 program cycle.

The continuation lines for the T8189ICF file specification define the following:

KNUM

Specifies the maximum number of devices to be acquired.

KINFDS

Specifies that the file information data structure is named FEEDBK. This structure redefines the I/O feedback areas. Refer to the *ICF Programmer's Guide* for a description of the I/O feedback areas.

KINFSR

Specifies the subroutine FAIL is to be called when a file exception condition occurs.

KID

Specifies that the device name for T8189ICF is in the field PGMDEV.

2

The STRCNV subroutine is called to start a conversation with the partner program. This is followed by the EXFMT operation, which allows the user to enter requests that are to be sent to the partner program.

3

The program loops until either F3 is pressed from the work station, which sets the indicator in the separate indicator area of the display file, or an error occurs in the transaction with the partner program.

4

The part number is sent to the partner program using a write operation. The write operation is issued using the ICF file record format ITEMRQ, which contains the confirm (CONFIRM keyword) and allow-write (ALWWRT keyword) functions. When these functions are used, the data is flushed, the data flow direction is changed from send to receive, and a confirmation request is sent to the partner program. The partner program must now respond with a positive or negative response.

5

If the partner program responds with a positive response (ICF return code of 0001) to the confirmation request, a read operation is issued using the ICF file record format ITEMDS to receive the part description. However, if the partner program responds with a negative response (ICF return code of 83C9) to the confirmation request, a read operation is issued using the ICF file record format ERRDES to receive the error message.

6

The EXFMT operation is issued to display the information returned by the partner program

and to obtain the next part number to be queried.

7

The following section of code performs the end-of-program processing. Indicator 85 determines if an error was detected.

8

If no error was detected (indicator 85 is set on), then a write operation is issued using the ICF file record format PGMEND, which contains the detach (DETACH keyword) function. The release operation (REL) is then issued to detach the program from the session.

9

If an error was detected (indicator 85 is not set), and the communications session is still active, a write operation is issued using the ICF file record format EOSREC, which contains the end-of-session (EOS keyword) function. The end-of-session function detaches the program from the session.

Note: If the end-of-session function is issued during an active transaction, APPC will end the session abnormally.

10

The last record indicator (LR) is set on. All files are implicitly closed, and the program ends.

11

The STRCNV subroutine establishes a conversation with the partner program.

12

The ICF00 program device is explicitly acquired using the ACQ operation. The acquire-program-device operation makes the program device available for input or output operations. A session is implicitly acquired for the work station when T8189DSP is opened.

Note: The program device ICF00 was previously added to the ICF file T8189ICF by the ADDICFDEVE command.

13

An evoke request is issued using a write operation. The write operation is issued using the ICF file record format PGMSTR, which contains the EVOKE, SECURITY, and SYNLVL keywords.

Note: On the EVOKE keyword, the library name is not specified. If the remote system

is an AS/400 system, the library list will be used to search for the program. Also, the remote program that is to be started can be any of the remote programs in this appendix and in Appendix F, "Example Programs for Applications Using the CPI Comm Application Program Interface" on page F-1.

14

The FAIL subroutine takes control when a file exception or error occurs. The FAIL subroutine handles all file exceptions or errors by passing control to the section of code that performs the end-of-program processing (**7**).

```

Compiler . . . . . : IBM AS/400 RPG/400
Command Options:
  Program . . . . . : APPCLIB/T8189IRS
  Source file . . . . . : QUSRTOOL/QATTRPG
  Source member . . . . . : *PGH
Text not available for message RXT0073 file QRPMSG.
  Generation options . . . . . : *NOLIST *NOXREF *NOATR *NODUMP *NOOPTIMIZE
  Source listing indentation . . . . . : *NONE
  SAA flagging . . . . . : *NOFLAG
  Generation severity level . . . . . : 29
  Print file . . . . . : *LIBL/QSYSVRT
  Replace program . . . . . : *YES
  Target release . . . . . : *CURRENT
  User profile . . . . . : *USER
  Authority . . . . . : *LIBCRTAUT
  Text . . . . . : *SRCMBRTXT
  Phase trace . . . . . : *NO
  Intermediate text dump . . . . . : *NONE
  Snap dump . . . . . : *NONE
  Codelist . . . . . : *NONE
  Ignore decimal data error . . . . . : *NO
Actual Program Source:
  Member . . . . . : T8189IRS
  File . . . . . : QATTRPG
  Library . . . . . : QUSRTOOL
  Last Change . . . . . : 09/26/90 08:27:43
SEQUENCE          IND   DO   LAST   PAGE   PROGRAM
NUMBER *...1....+...2....+...3....+...4....+...5....+...6....+...7...*  USE   NUM   UPDATE  LINE   ID
          Source Listing
10 H
1 20 FT8189DSPCF E          WORKSTN
   RECORD FORMAT(S): LIBRARY APPCLIB FILE T8189DSP.
   EXTERNAL FORMAT PROMPT RPG NAME PROMPT
30 FT8189ICFCF E          WORKSTN
40 F                                KNUM      1
50 F                                KINFSR   FAIL
60 F                                KINFDS  FEEDBK
70 F                                KID     PGMDEV
   RECORD FORMAT(S): LIBRARY APPCLIB FILE T8189ICF.
   EXTERNAL FORMAT PGMSTR RPG NAME PGMSTR
   EXTERNAL FORMAT ITEMRQ RPG NAME ITEMRQ
   EXTERNAL FORMAT ITEMDS RPG NAME ITEMDS
   EXTERNAL FORMAT ERRDES RPG NAME ERRDES
   EXTERNAL FORMAT PGMEND RPG NAME PGMEND
   EXTERNAL FORMAT EOSREC RPG NAME EOSREC
   EXTERNAL FORMAT PGMERR RPG NAME PGMERR
   EXTERNAL FORMAT ITEMOK RPG NAME ITEMOK
A000000 INPUT FIELDS FOR RECORD PROMPT FILE T8189DSP FORMAT PROMPT.
A000001                                1 5 PARTN
B000000 INPUT FIELDS FOR RECORD PGMSTR FILE T8189ICF FORMAT PGMSTR.
C000000 INPUT FIELDS FOR RECORD ITEMRQ FILE T8189ICF FORMAT ITEMRQ.
C000001                                1 5 PARTNM
D000000 INPUT FIELDS FOR RECORD ITEMDS FILE T8189ICF FORMAT ITEMDS.
D000001                                1 25 PARTDS
E000000 INPUT FIELDS FOR RECORD ERRDES FILE T8189ICF FORMAT ERRDES.
E000001                                1 40 ERROR
F000000 INPUT FIELDS FOR RECORD PGMEND FILE T8189ICF FORMAT PGMEND.
G000000 INPUT FIELDS FOR RECORD EOSREC FILE T8189ICF FORMAT EOSREC.
H000000 INPUT FIELDS FOR RECORD PGMERR FILE T8189ICF FORMAT PGMERR.
I000000 INPUT FIELDS FOR RECORD ITEMOK FILE T8189ICF FORMAT ITEMOK.
80 IFEBBK DS
90 I                                B 372 3750ACTLEN
100 I                                401 404 MAJMIN
110 I                                401 402 MAJCOD
120 I                                403 404 MINCOD

```

Figure E-9 (Part 1 of 6). RPG/400 Inquiry Example – Local Program

```

130 I*****
140 I* Program name.....: T8189IRS          *
150 I* Program description.: ICF local program *
160 I* Language.....: RPG/400              *
170 I*                                     *
180 I* This program invokes a program to handle part inquiry on *
190 I* the remote system. The acquire operation is used to *
200 I* establish a communications session. A write operation *
210 I* with the evoke function is then issued, which results in *
220 I* the establishment of a conversation with the remote *
SEQUENCE NUMBER *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  IND DO LAST PAGE PROGRAM
NUMBER USE NUM UPDATE LINE ID
230 I* program. A display which prompts the user for the part *
240 I* number for which part information is requested is then *
250 I* displayed. When the user presses Enter, a write operation *
260 I* is issued (the data sent to the partner program is the *
270 I* part number). Note that the write operation was issued *
280 I* with the confirm (CONFIRM) and allow-write (ALWVRT) *
290 I* functions. These functions results in the flushing of the *
300 I* data (to be sent to the partner program), the changing of *
310 I* the data flow direction (the partner program can send the *
320 I* response), and the sending of a confirmation request to *
330 I* the partner program. If the partner program responds *
340 I* with a positive response to the confirmation request (using *
350 I* the RSPCONFIRM function), the ICF return code on the *
360 I* write operation will be set to 0001 (indicating that *
370 I* the part number was found); a read operation is then *
380 I* issued to receive the part description. However, if *
390 I* the partner program responds with a negative response *
400 I* to the confirmation request (using the FAIL function), *
410 I* the ICF return code on the write operation will be set *
420 I* to 83C9 (indicating that the part number was not found); *
430 I* a read operation is issued to receive the error message. *
440 I* *
450 I* The error message or part description (depending on *
460 I* whether the part number was found) will be displayed on *
470 I* the screen. *
480 I* *
490 I* This program will continue to handle inquiries until the *
500 I* user presses the F3=Exit key. When F3=Exit is pressed, *
510 I* a write operation with the detach (DETACH) function *
520 I* is issued to end the conversation, and program processing *
530 I* ends. *
540 I* *
550 I* NOTE 1: If an unexpected ICF return code is received on *
560 I* any of the read or write operations, the *
570 I* program will abnormally end the conversation (if *
580 I* it is still active), and program processing will *
590 I* end. *
600 I* *
610 I* NOTE 2: On the receive operation, if the actual received *
620 I* data length (obtained from the I/O feedback area) *
630 I* does not match what was expected, or if the *
640 I* ICF return code is not 0000 (indication that *
650 I* the partner program is ready to receive data), the *
660 I* program will abnormally end the conversation (if *
670 I* it is still active), and program processing will *
680 I* end. *
690 I* *
700 I* NOTE 3: This program can start ANY of the "remote" *
710 I* program examples in the APPC Programmer's *
720 I* Guide by changing the PGMID variable to the *
730 I* remote program that is to be started. *
740 I* *
750 I*****
760 I*

```

Figure E-9 (Part 2 of 6). RPG/400 Inquiry Example – Local Program

SEQUENCE NUMBER	*...1...+...2...+...3...+...4...+...5...+...6...+...7...*	IND USE	DO NUM	LAST UPDATE	PAGE LINE	PROGRAM ID
770	ILCLVAR DS					
780	I B 1 40REQLN					
790	C*					
800	C*****					
810	C* START OF PROGRAM					*
820	C*					*
830	C* Files are implicitly opened, a conversation with the					*
840	C* remote program is started, and the part inquiry					*
850	C* screen is displayed. Inquiries are handled until					*
860	C* the user presses the F3=Exit key, in which case					*
870	C* the conversation will be ended and the program will end.					*
880	C*****					
890	C*					
900	C 2 EXSR STRCNV					
910	C EXFMPROMPT					
920	C*					
930	C 3 *IN99 DOWEQ'0'			B001		
940	C*					
950	C*****					
960	C* The part number that the user has requested information					*
970	C* for is sent to the remote program using the write					*
980	C* operation with the confirm and allow-write functions.					*
990	C*****					
1000	C*					
1010	C MOVE PARTN PARTNM			001		
1020	C 4 WRITEITEMRQ 88		2	001		
1030	C*					
1040	C*****					
1050	C* The read operation is issued to receive the response					*
1060	C* from the remote program (the response can either be					*
1070	C* an error message or the part description, depending					*
1080	C* on whether the part was found or not).					*
1090	C*****					
1100	C*					
1110	C 5 MAJMIN IFEQ '0001'			B002		
1120	C Z-ADD25 REQLN			002		
1130	C READ ITEMDS 88		3	002		
1140	C MOVE PARTDS PARTD			002		
1150	C MOVE *BLANKS ERRORL			002		
1160	C ELSE X002					
1170	C MAJMIN IFEQ '83C9'			B003		
1180	C Z-ADD40 REQLN			003		
1190	C READ ERRDES 88		3	003		
1200	C MOVE ERRORL ERRORL			003		
1210	C MOVE *BLANKS PARTD			003		
1220	C ELSE X003					
1230	C GOTO ENDCNV			003		
1240	C END E003					
1250	C END E002					
1260	C*					
1270	C MAJMIN IFNE '0000'			B002		
1280	C ACTLEN ORNE REQLN			002		
1290	C GOTO ENDCNV			002		
1300	C END E002					
1310	C*					
1320	C 6 EXFMPROMPT			001		
1330	C*					
1340	C END E001					
1350	C*					
1360	C SETON 85		1			
1370	C*					

Figure E-9 (Part 3 of 6). RPG/400 Inquiry Example – Local Program

```

1380 C*****
1390 C* The following code handles the end-of-program processing. *
1400 C* This includes the ending of the conversation with *
1410 C* the remote system (if conversation is active) by either *
1420 C* issuing a write operation with the detach function *
1430 C* followed by a release operation (for non-error conditions),*
1440 C* or by issuing a write operation with the end-of-session *
1450 C* function (for error conditions). The last record *
1460 C* indicator is then set on to end the program. *
1470 C*****
1480 C*
1490 C 7 ENDCNV TAG
1500 C*
1510 C MAJCOD IFNE '80' B001
1520 C MAJCOD ANDNE'81' 001
1530 C MAJCOD ANDNE'82' 001
1540 C*****
1550 C* Indicator 85 (set previously) determines if the *
1560 C* conversation is to end normally or abnormally. If the *
1570 C* indicator is set on, conversation will be ended normally; *
1580 C* otherwise, conversation will be ended abnormally. *
1590 C*****
1600 C 8 85 WRITEPGMEND 88 2 001
1610 C 85 'ICF00' REL T8189ICF 001
1620 C*
1630 C 9 N85 WRITEEOSREC 88 2 001
1640 C END E001
1650 C*
1660 C ENDPGM TAG
1670 C 10 SETON LR 1
1680 C*
1690 C*****
1700 C* "STRCNV" subroutine. *
1710 C* *
1720 C* This subroutine establishes a conversation with the *
1730 C* remote program. *
1740 C*****
1750 C*
1760 CSR 11 STRCNV BEGSR
1770 C*
1780 C*****
1790 C* The acquire operation is issued. *
1800 C*****
1810 C*
1820 CSR 12 'ICF00' ACQ T8189ICF
SEQUENCE *...1....2....3....4....5....6....7...* IND DO LAST PAGE PROGRAM
NUMBER USE NUM UPDATE LINE ID
1830 CSR MAJMIN CABNE'0000' ENDPGM
1840 CSR MOVEL'ICF00' PGMDEV 10
1850 C*
1860 C*****
1870 C* A write operation with the evoke function is issued so *
1880 C* that a conversation can be started. *
1890 C*****
1900 C*
1910 CSR MOVEL'T8189IRT'PGMID
1920 CSR 13 WRITEPGMSTR 88 2
1930 CSR MAJMIN CABNE'0000' ENDCNV
1940 C*
1950 CSR ENDSR
1960 C*
1970 C*****
1980 C* "FAIL" subroutine. *
1990 C* *
2000 C* This subroutine handles file exception/errors. *
2010 C*****
2020 C*
2030 CSR 14 FAIL BEGSR
2040 CSR GOTO ENDCNV
2050 CSR ENDSR

```

Figure E-9 (Part 4 of 6). RPG/400 Inquiry Example – Local Program

```

J000000 OUTPUT FIELDS FOR RECORD PROMPT FILE T8189DSP FORMAT PROMPT.
J000001          PARTN      5 CHAR      5
J000002          PARTD     30 CHAR     25
J000003          ERRORL    70 CHAR     40
K000000 OUTPUT FIELDS FOR RECORD PGMSTR FILE T8189ICF FORMAT PGMSTR.
K000001          PGMID     10 CHAR     10
L000000 OUTPUT FIELDS FOR RECORD ITEMRQ FILE T8189ICF FORMAT ITEMRQ.
L000001          PARTNM     5 CHAR      5
M000000 OUTPUT FIELDS FOR RECORD PGMEND FILE T8189ICF FORMAT PGMEND.
N000000 OUTPUT FIELDS FOR RECORD EOSREC FILE T8189ICF FORMAT EOSREC.
***** END OF SOURCE *****
      Additional Diagnostic Messages
* 7111          SOURCE FILE MEMBER HAS AN UNEXPECTED SOURCE TYPE.
* 7089          20  RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE T8189DSP.
* 7089          30  RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE T8189ICF.
      Cross Reference
File and Record References:
      FILE/RCD  DEV/RCD      REFERENCES (D=DEFINED)
01 T8189DSP    WORKSTN      20D
      PROMPT          20D A000000      910      1320 J000000
02 T8189ICF    WORKSTN      30D 1610      1820
      EOSREC          30D G000000      1630 N000000
      ERRDES          30D E000000      1190
      ITEMSD          30D D000000      1130
      ITEMOK          30D I000000
      ITEMRQ          30D C000000      1020 L000000
      PGMEND          30D F000000      1600 M000000
      PGMERR          30D H000000
      PGMSTR          30D B000000      1920 K000000
Field References:
      FIELD      ATTR      REFERENCES (M=MODIFIED D=DEFINED)
*IN99          A(1)        930
ACTLEN         B(9,0)      900 1280
ENDCNV         TAG         1230 1290 1490D 1930 2040
ENDPGM         TAG         1660D 1830
ERRORD         A(40)      E000001D 1200
ERRORL         A(40)      1150M 1200M J000003D
FAIL           BEGSR       30 2030D
FEEDBK         DS(404)     30 80D
* 7031 LCLVAR   DS(4)       770D
MAJCOD         A(2)        110D 1510 1520 1530
MAJMIN         A(4)        100D 1110 1170 1270 1830
              1930
* 7031 MINCOD   A(2)        120D
PARTD          A(25)       1140M 1210M J000002D
PARTDS         A(25)       D000001D 1140
PARTN          A(5)        A000001D 1010 J000001D
PARTNM         A(5)        C000001D 1010M L000001D
PGMDEV         A(10)       1840D
PGMID          A(10)       1910M K000001D
REQLEN         B(9,0)      780D 1120M 1180M 1280
STRCNV         BEGSR       900 1760D
*BLANKS        LITERAL    1150 1210
'ICF00'        LITERAL    1610 1820 1840
'T8189IRT'     LITERAL    1910
'0'            LITERAL    930
'0000'         LITERAL    1270 1830 1930
'0001'         LITERAL    1110
'80'           LITERAL    1510
'81'           LITERAL    1520

```

Figure E-9 (Part 5 of 6). RPG/400 Inquiry Example – Local Program

```

      '82'      LITERAL  1530
      '83C9'   LITERAL  1170
      25       LITERAL  1120
      40       LITERAL  1180
Indicator References:
INDICATOR REFERENCES (M=MODIFIED D=DEFINED)
*IN          930
LR          1670M
85          1360M  1600  1610  1630
* 7031 88    1020M  1130M  1190M  1600M  1630M  1920M
      99      930
      * * * * * E N D   O F   C R O S S   R E F E R E N C E   * * * * *
                          M e s s a g e   S u m m a r y
* QRG7031 Severity: 00  Number: 3
      Message . . . . : The Name or indicator is not referenced.
* QRG7089 Severity: 00  Number: 2
      Message . . . . : The RPG provides Separate-Indicator area for
      file.
* QRG7111 Severity: 00  Number: 1
      Message . . . . : Unexpected source type.
      * * * * * E N D   O F   M E S S A G E   S U M M A R Y   * * * * *
                          F i n a l   S u m m a r y
Message Count: (by Severity Number)
      TOTAL  00  10  20  30  40  50
      6      6   0   0   0   0   0
Program Source Totals:
Records . . . . . : 205
Specifications . . . . . : 62
Table Records . . . . . : 0
Comments . . . . . : 143
PRM has been called.
Program T8189IRS is placed in library APPCLIB. 00 highest Error-Severity-Code.
      * * * * * E N D   O F   C O M P I L A T I O N   * * * * *

```

Figure E-9 (Part 6 of 6). RPG/400 Inquiry Example – Local Program

RPG/400 Remote Program for Inquiry Application (Example 3)

The following explains the structure of the RPG/400 remote program that handles requests sent by the partner program.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure E-10 on page E-46.

Note: On any type of error that is not expected (for example, open errors ICF return code on an I/O operation), the session is ended and the program ends.

1

The file specification defines the files used in the program.

T8189ICF is the ICF file used to receive records from, and send records to, the partner program. T8189ICF uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

T8189DB is the database file that contains the valid part numbers and part descriptions.

All files are implicitly opened at the beginning of the RPG/400 program cycle.

The continuation lines for the T8189ICF file specification define the following:

KNUM

Specifies the maximum number of devices to be acquired.

KINFDS

Specifies that the file information data structure is named FEEDBK. This structure redefines the I/O feedback areas.

KINFSR

Specifies the subroutine FAIL is to be called when a file exception condition occurs.

2

The STRCNV and GETDTA subroutines are called to start a conversation with the partner program and wait on a request by the partner program, respectively.

3

The program loops until there are no more requests to process, or an error occurs in the transaction with the partner program.

4

A search of the database file is performed using the part number received from the partner program as the key.

Note: If the part number is not found, indicator 98 is set on.

5

If indicator 98 is set on (indicating that the part number was not found in the database file), a write operation is issued using the ICF file record format PGMERR, which contains the fail (FAIL keyword) function. As a result, a negative response to the received confirmation request is sent to the partner program. This is followed by a second write operation using the ICF file record format ERRDES, which contains the error message to be sent and the allow-write (ALWWRT keyword) function. When the allow-write function is used, the data is flushed and the data flow direction is changed.

If indicator 98 is not set on (indicating that the part number was found in the database file), a write operation is issued using the ICF file record format ITEMOK, which contains the respond-to-confirm (RSPCONFIRM keyword) function. As a result, a positive response to the received confirmation request is sent to the partner program. This is followed by a second write operation using the ICF file record format ITEMDS, which contains the requested information to be sent and the allow-write function.

6

The GETDTA subroutine is called to wait on a request by the partner program.

7

The following section of code performs the end-of-program processing. Whether or not a detach indication is received determines if an error was detected.

8

If the ICF return code is 0308, indicating that a detach was received, a release operation

(REL) is issued to detach the program from the session.

9

If a detach indication was not received and the communications session is still active, a write operation is issued using the ICF file record format EOSREC, which contains the end-of-session (EOS keyword) function. The end-of-session function detaches the program from the session.

Note: If the end-of-session function is issued during an active transaction, APPC will end the session abnormally.

10

The last record indicator (LR) is set on. All files are implicitly closed, and the program ends.

11

The STRCNV subroutine establishes a conversation with the partner program by explicitly acquiring the ICF01 program device using the ACQ operation.

Note: The program device ICF01 was previously added to the ICF file T8189ICF by the ADDICFDEVE command.

12

The GETDTA subroutine waits for a request from the partner program by issuing a read operation using the ICF file record format ITEMRQ.

Note: A transaction is processed if data is received with a turnaround indication, and the partner program requested confirmation. This is indicated by the ICF return code 0014.

13

The FAIL subroutine takes control when a file exception or error occurs. The FAIL subroutine handles all file exceptions or errors by passing control to the section of code that performs the end-of-program processing (**7**).

Compiler : IBM AS/400 RPG/400

Command Options:

```

Program . . . . . : APPCLIB/T8189IRT
Source file . . . . . : QUSRTOOL/QATTRPG
Source member . . . . . : *PGM
Text not available for message RXT0073 file QRPMSG.
Generation options . . . . . : *NOLIST *NOXREF *NOATR *NODUMP *NOOPTIMIZE
Source listing indentation . . . . . : *NONE
SAA flagging . . . . . : *NOFLAG
Generation severity level . . . . . : 29
Print file . . . . . : *LIBL/QSYSPRT
Replace program . . . . . : *YES
Target release . . . . . : *CURRENT
User profile . . . . . : *USER
Authority . . . . . : *LIBCRTAUT
Text . . . . . : *SRCMBRTXT
Phase trace . . . . . : *NO
Intermediate text dump . . . . . : *NONE
Snap dump . . . . . : *NONE
Codelist . . . . . : *NONE
Ignore decimal data error . . . . . : *NO
    
```

Actual Program Source:

```

Member . . . . . : T8189IRT
File . . . . . : QATTRPG
Library . . . . . : QUSRTOOL
Last Change . . . . . : 09/26/90 08:27:45
    
```

SEQUENCE	IND	DO	LAST	PAGE	PROGRAM
NUMBER	USE	NUM	UPDATE	LINE	ID

```

          S o u r c e   L i s t i n g
10 H
1 FT8189DB IF E K DISK
   RECORD FORMAT(S): LIBRARY APPCLIB FILE T8189DB.
   EXTERNAL FORMAT DBRCD RPG NAME DBRCD
30 FT8189ICFCF E WORKSTN
40 F KNUM 1
50 F KINFSR FAIL
60 F KINFDS FEEDBK
   RECORD FORMAT(S): LIBRARY APPCLIB FILE T8189ICF.
   EXTERNAL FORMAT PGMSTR RPG NAME PGMSTR
   EXTERNAL FORMAT ITEMRQ RPG NAME ITEMRQ
   EXTERNAL FORMAT ITEMDS RPG NAME ITEMDS
   EXTERNAL FORMAT ERRDES RPG NAME ERRDES
   EXTERNAL FORMAT PGMEND RPG NAME PGMEND
   EXTERNAL FORMAT EOSREC RPG NAME EOSREC
   EXTERNAL FORMAT PGMERR RPG NAME PGMERR
   EXTERNAL FORMAT ITEMOK RPG NAME ITEMOK
70 E MSG 1 1 40
A000000 INPUT FIELDS FOR RECORD DBRCD FILE T8189DB FORMAT DBRCD.
A000001 1 5 ITEMNH
A000002 6 30 ITEMN
B000000 INPUT FIELDS FOR RECORD PGMSTR FILE T8189ICF FORMAT PGMSTR.
C000000 INPUT FIELDS FOR RECORD ITEMRQ FILE T8189ICF FORMAT ITEMRQ.
C000001 1 5 PARTNH
D000000 INPUT FIELDS FOR RECORD ITEMDS FILE T8189ICF FORMAT ITEMDS.
D000001 1 25 PARTDS
E000000 INPUT FIELDS FOR RECORD ERRDES FILE T8189ICF FORMAT ERRDES.
E000001 1 40 ERRORN
F000000 INPUT FIELDS FOR RECORD PGMEND FILE T8189ICF FORMAT PGMEND.
G000000 INPUT FIELDS FOR RECORD EOSREC FILE T8189ICF FORMAT EOSREC.
H000000 INPUT FIELDS FOR RECORD PGMERR FILE T8189ICF FORMAT PGMERR.
I000000 INPUT FIELDS FOR RECORD ITEMOK FILE T8189ICF FORMAT ITEMOK.
80 IFEEBKB DS
90 I B 372 375OACTLEN
100 I 401 404 MAJMIN
110 I 401 402 MAJCOD
120 I 403 404 MINCOD
    
```

Figure E-10 (Part 1 of 5). RPG/400 Inquiry Example – Remote Program

```

130 I*****
140 I* Program name.....: T8189IRT          *
150 I* Program description.: ICF remote program      *
160 I* Language.....: RPG/400                *
170 I*                                          *
180 I* This program accepts the incoming conversation by issuing *
190 I* the acquire operation to acquire the requesting program *
5738RG1 V2R1M0 910329          IBM AS/400 RPG/400          APPCLIB/T8189IRT          09/26/90 08:34:48          Page 3
SEQUENCE                               IND   DO   LAST   PAGE   PROGRAM
NUMBER  *...1....+...2....+...3....+...4....+...5....+...6....+...7...*  USE   NUM  UPDATE  LINE   ID
200 I* device. A read operation is then issued to receive the *
210 I* part number from the remote system. The read operation *
220 I* completes with an ICF return code of 0014, indicating *
230 I* that data with a turnaround indication was received, and *
240 I* that the partner program also requested confirmation. The *
250 I* database file T8189DB is searched for the received part *
260 I* number. If the part number is found, a write operation *
270 I* with the respond-to-confirm (RSPCONFIRM) function is *
280 I* issued, followed by a write operation containing the *
290 I* part description corresponding to the part number *
300 I* retrieved from the database file. However, if the part *
310 I* number is not found, a write operation with the *
320 I* negative-response (FAIL) function is issued, followed by *
330 I* a write operation containing an error message describing *
340 I* the error. The write operation sending either the part *
350 I* description or the error message is issued with the *
360 I* allow-write (ALMWRT) function. Using the allow-write *
370 I* function results in the flushing of the data and the *
380 I* changing of the data flow direction. The partner program *
390 I* can send more inquiries. *
400 I* *
410 I* This program will continue to handle inquiries from the *
420 I* partner program until a detach indication is received. *
430 I* Then the program ends. *
440 I* *
450 I* NOTE 1: If an unexpected ICF return code is received on *
460 I* any of the read or write operations, the *
470 I* program will abnormally end the conversation (if *
480 I* it is still active), and program processing will *
490 I* end. *
500 I* *
510 I* NOTE 2: On the receive operation, if the actual received *
520 I* data length (obtained from the I/O feedback area) *
530 I* does not match what was expected, or if the *
540 I* ICF return code is not 0014 (indication that *
550 I* data was received with a turnaround indicator, *
560 I* and partner program requested confirmation), the *
570 I* program will abnormally end the conversation (if *
580 I* it is still active), and program processing will *
590 I* end. *
600 I* *
610 I* NOTE 3: This program can be started by ANY of the *
620 I* "local" program examples in the APPC Programmer's *
630 I* Guide. *
640 I* *
650 I*****
660 I*
670 C*****
680 C* START OF PROGRAM *
690 C* *
700 C* Files are implicitly opened, a conversation with the *
710 C* remote program is started, and the part inquiry *
5738RG1 V2R1M0 910329          IBM AS/400 RPG/400          APPCLIB/T8189IRT          09/26/90 08:34:48          Page 4
SEQUENCE                               IND   DO   LAST   PAGE   PROGRAM
NUMBER  *...1....+...2....+...3....+...4....+...5....+...6....+...7...*  USE   NUM  UPDATE  LINE   ID
720 C* processing starts. Inquiries are handled until a *
730 C* detach indication is received. *
740 C*****
750 C*
760 C  2          EXSR STRCNV
770 C          EXSR GETDTA

```

Figure E-10 (Part 2 of 5). RPG/400 Inquiry Example – Remote Program

```

780 C*
790 C 3 MAJMIN DOWEQ'0014' B001
800 C MOVE PARTNM ITEMNM 001
810 C*
820 C*****
830 C* A search of the database file is done using the part *
840 C* number as the key (indicator 98 is set on if the part *
850 C* number is not found). *
860 C*****
870 C*
880 C 4 ITEMNM CHAINDBRCD 98 1 001
890 C*
900 C*****
910 C* If the part number is not found, a write operation with *
920 C* the FAIL function is issued, and then a write operation *
930 C* with the error message is issued. Otherwise, a positive *
940 C* response to the confirmation request is issued, and then *
950 C* a write operation with the part description is issued. *
960 C*****
970 C*
980 C 98 5 WRITEPGMERR 001
990 C*
1000 C N98 WRITEITEHOK 001
1010 C MAJMIN CABNE'0000' ENDCNV 001
1020 C 98 MOVEHMSG,1 ERROR 001
1030 C 98 WRITEERRDES 001
1040 C N98 MOVELITEHD PARTDS 001
1050 C N98 WRITEITEMDS 001
1060 C MAJMIN CABNE'0001' ENDCNV 001
1070 C*
1080 C 6 EXSR GETDTA 001
1090 C END E001
1100 C*
1110 C*****
1120 C* The following code handles the end-of-program processing. *
1130 C* This includes the ending of the conversation with *
1140 C* the remote system (if conversation is active) by either *
1150 C* issuing a release operation (for non-error conditions), *
1160 C* or by issuing a write operation with the end-of-session *
1170 C* function (for error conditions). The last record *
1180 C* indicator is then set on to end the program. *
1190 C*****
1200 C*
1210 C 7 ENDCNV TAG
1220 C*
1230 C MAJCOD IFNE '80' B001
1240 C MAJCOD ANDNE '81' 001
1250 C MAJCOD ANDNE '82' 001
5738RG1 V2R1M0 910329 IBH AS/400 RPG/400 APPCLIB/T8189IRT 09/26/90 08:34:48 Page 5
SEQUENCE NUMBER *...1....+...2....+...3....+...4....+...5....+...6....+...7...* IND DO LAST PAGE PROGRAM
USE NUM UPDATE LINE ID
1260 C 8 MAJMIN IFEQ '0308' B002
1270 C 'ICF01' REL T8189ICF 002
1280 C ELSE X002
1290 C 9 WRITEEOSREC 88 2 002
1300 C END E002
1310 C END E001
1320 C*
1330 C ENDPGM TAG
1340 C 10 SETON LR 1
1350 C*
1360 C*****
1370 C* "STRCNV" subroutine. *
1380 C* *
1390 C* This subroutine establishes a conversation with the *
1400 C* remote program. *
1410 C*****
1420 C*
1430 CSR 11 STRCNV BEGSR
1440 C*

```

Figure E-10 (Part 3 of 5). RPG/400 Inquiry Example — Remote Program

```

1450 C*****
1460 C* The acquire operation is issued. *
1470 C*****
1480 C*
1490 CSR      'ICF01'  ACQ  T8189ICF
1500 CSR      MAJMIN  CABNE'0000'  ENDPGM
1510 C*
1520 CSR      ENDSR
1530 C*
1540 C*****
1550 C* "GETDTA" subroutine. *
1560 C* *
1570 C* This subroutine waits for incoming data from the partner *
1580 C* program by issuing a read operation. *
1590 C*****
1600 C*
1610 CSR      12  GETDTA  BEGSR
1620 C*
1630 CSR      READ ITEMRQ          88          3
1640 C*
1650 CSR      MAJMIN  IFEQ '0014'          B001
1660 CSR      ACTLEN  IFNE 5          B002
1670 CSR      GOTO ENDCNV          002
1680 CSR      END          E002
1690 CSR      END          E001
1700 CSR      ENDSR
1710 C*
1720 C*****
1730 C* "FAIL" subroutine. *
1740 C* *
1750 C* This subroutine handles file exception/errors. *
1760 C*****
1770 C*
1780 CSR      13  FAIL  BEGSR
1790 CSR      GOTO ENDCNV
5738RG1 V2R1M0 910329 IBM AS/400 RPG/400 APPCLIB/T8189IRT 09/26/90 08:34:48 Page 6
SEQUENCE IND DO LAST PAGE PROGRAM
NUMBER *...1...+...2...+...3...+...4...+...5...+...6...+...7...* USE NUM UPDATE LINE ID
1800 CSR ENDSR
J000000 OUTPUT FIELDS FOR RECORD ITEMDS FILE T8189ICF FORMAT ITEMDS.
J000001 PARTDS 25 CHAR 25
K000000 OUTPUT FIELDS FOR RECORD ERDES FILE T8189ICF FORMAT ERDES.
K000001 ERROR 40 CHAR 40
L000000 OUTPUT FIELDS FOR RECORD EOSREC FILE T8189ICF FORMAT EOSREC.
M000000 OUTPUT FIELDS FOR RECORD PGMERR FILE T8189ICF FORMAT PGMERR.
N000000 OUTPUT FIELDS FOR RECORD ITEMOK FILE T8189ICF FORMAT ITEMOK.
***** END OF SOURCE *****
Additional Diagnostic Messages
* 7111 SOURCE FILE MEMBER HAS AN UNEXPECTED SOURCE TYPE.
* 7089 30 RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE T8189ICF.
5738RG1 V2R1M0 910329 IBM AS/400 RPG/400 APPCLIB/T8189IRT 09/26/90 08:34:48 Page 7
SEQUENCE LAST
NUMBER *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
Compile - Time Tables
Table/Array . . . . . : MSG
1820 The requested part was not found.
5738RG1 V2R1M0 910329 IBM AS/400 RPG/400 APPCLIB/T8189IRT 09/26/90 08:34:48 Page 8
Key Field Information
PHYSICAL LOGICAL
FILE/RCD FIELD FIELD ATTRIBUTES
01 T8189DB
DBRCD
ITEMNH CHAR 5
5738RG1 V2R1M0 910329 IBM AS/400 RPG/400 APPCLIB/T8189IRT 09/26/90 08:34:48 Page 9
Cross Reference
File and Record References:
FILE/RCD DEV/RCD REFERENCES (D=DEFINED)
01 T8189DB DISK 20D
DBRCD 20D A000000 880
02 T8189ICF WORKSTN 30D 1270 1490

```

Figure E-10 (Part 4 of 5). RPG/400 Inquiry Example – Remote Program

```

EOSREC          30D G000000    1290 L000000
ERRDES          30D E000000    1030 K000000
ITEMDS          30D D000000    1050 J000000
ITEMOK          30D I000000    1000 N000000
ITEMRQ          30D C000000    1630
PGMEND          30D F000000
PGMERR          30D H000000     980 M000000
PGMSTR          30D B000000

```

Field References:

```

FIELD          ATTR          REFERENCES (M=MODIFIED D=DEFINED)
ACTLEN         B(9,0)          90D    1660
ENDCNV         TAG            1010   1060   1210D  1670   1790
ENDPGM         TAG            1330D  1500
ERROR          A(40)          E000001D 1020M K000001D
FAIL           BEGSR          30     1780D
FEEDBK         DS(404)         30     80D
GETDTA         BEGSR          770    1080   1610D
ITEMD          A(25)          A000002D 1040
ITEMNM         A(5)            A000001D 800M    880
MAJCOD         A(2)            110D   1230   1240   1250
MAJMIN         A(4)            100D   790    1010   1060   1260
                1500    1650
* 7031 MINCOD   A(2)            120D
MSG(1)         A(40)            70D
MSG,1          1020
PARTDS         A(25)          D000001D 1040M J000001D
PARTNM         A(5)            C000001D 800
STRCNV         BEGSR          760    1430D
'ICF01'        LITERAL          1270   1490
'0000'         LITERAL          1010   1500
'0001'         LITERAL          1060
'0008'         LITERAL          1260
'0014'         LITERAL          790    1650
'80'           LITERAL          1230
'81'           LITERAL          1240
'82'           LITERAL          1250
1              LITERAL          1020
5              LITERAL          1660

```

5738RG1 V2R1M0 910329 IBM AS/400 RPG/400 APPCLIB/T8189IRT 09/26/90 08:34:48 Page 10

Indicator References:

```

INDICATOR      REFERENCES (M=MODIFIED D=DEFINED)
LR              1340M
* 7031 88       1290M  1630M
          98       880M   980    1000   1020   1030   1040
                1050

```

***** END OF CROSS REFERENCE *****

5738RG1 V2R1M0 910329 IBM AS/400 RPG/400 APPCLIB/T8189IRT 09/26/90 08:34:48 Page 11

Message Summary

```

* QRG7031 Severity: 00 Number: 2
Message . . . . : The Name or indicator is not referenced.
* QRG7089 Severity: 00 Number: 1
Message . . . . : The RPG provides Separate-Indicator area for
file.
* QRG7111 Severity: 00 Number: 1
Message . . . . : Unexpected source type.

```

***** END OF MESSAGE SUMMARY *****

5738RG1 V2R1M0 910329 IBM AS/400 RPG/400 APPCLIB/T8189IRT 09/26/90 08:34:48 Page 12

Final Summary

```

Message Count: (by Severity Number)
TOTAL  00  10  20  30  40  50
        4   4   0   0   0   0

```

Program Source Totals:

```

Records . . . . . : 182
Specifications . . . . . : 54
Table Records . . . . . : 1
Comments . . . . . : 126

```

PRM has been called.

Program T8189IRT is placed in library APPCLIB. 00 highest Error-Severity-Code.

***** END OF COMPILATION *****

Figure E-10 (Part 5 of 5). RPG/400 Inquiry Example — Remote Program

Appendix F. Example Programs for Applications Using the CPI Comm Application Program Interface

This appendix provides example programs which demonstrate how to use the APPC support on the AS/400 system using CPI Communications. The example programs included in this appendix are also available in the QUSRTOOL library (see file QATTINFO, member T8189INF in library QUSRTOOL).

The following example programs are provided:

- Example 1 (starting on page F-2) shows two C/400 programs.
- Example 2 (starting on page F-18) shows two COBOL/400 programs.
- Example 3 (starting on page F-35) shows two RPG/400 programs.

Each example contains two programs: the local program, which initiates the transaction, and the remote program, which performs services relating to the processing of the transaction.

Note: The term **remote program** is used in the following examples to refer to the program with which the local program communicates, even though the remote program may not be on a remote system. Similarly, the term **remote system** may actually be the same system in which the local program resides.

Figure F-1 illustrates the environment in which the example programs run. The local program requests the entry of a part number from the display station. That part number is then transmitted to the partner program, where a database file is searched. If the number is found, the partner program responds with a positive response followed by the requested information. If the part number is not found, the partner program responds with a negative response followed by an error message.

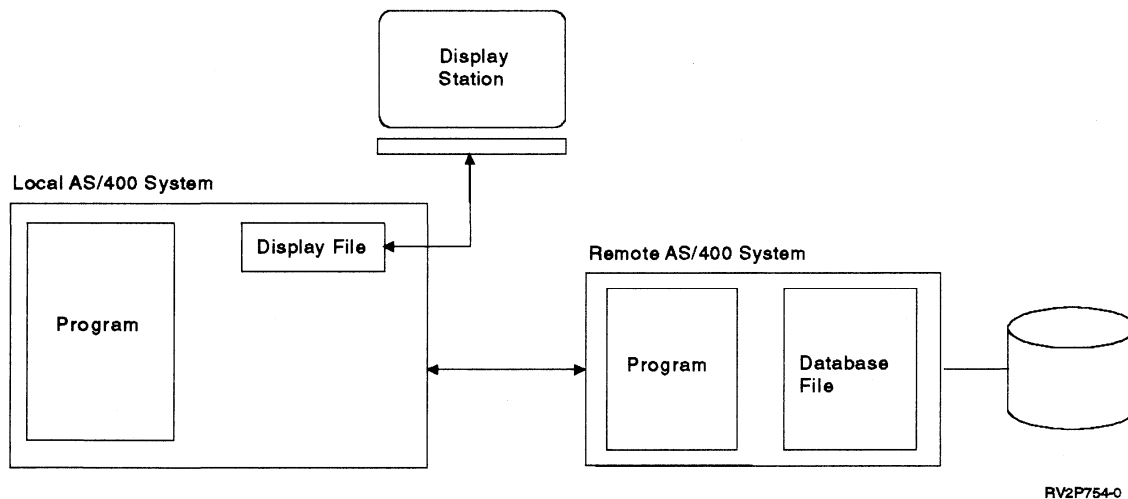


Figure F-1. Inquiry Example

Objects Used by Program Examples

The following objects are used by the program examples:

- Communications side information, T8189CSI
- Display file, T8189DSP
- Database file, T8189DB

Communications Side Information Object (T8189CSI)

In this example the side information object is used by the local program to provide initialization information for the conversation characteristics, such as the *partner_LU_name*, *mode_name*, and *TP_name*. The command used to create the side information object used by the local programs is

```

CRTCSI CSI(APPCLIB/T8189CSI) RMTLOCNAME(T8189LA) TNSPGM(TP_NAME)
MODE(BLANK) RMTNETID(*NETATR)
TEXT('Side information object for APPC examples')

```

Note: The side information object is not required. You can specify a symbolic destination name (*sym_dest_name*) of blanks on the Initialize_Conversation (CMINIT) call. You must then use the appropriate Set calls to set the following conversation characteristics: *partner_LU_name*, *mode_name*, and *TP_name*. For the following examples, you would need to add source code in the local programs that would use the Set_Partner_LU_Name (CMSPLN) call to set the *partner_LU_name* to T8189LA. The default *mode_name* that is used when a *sym_dest_name* of blanks is used is BLANK, and the Set_TP_Name (CMSTPN) call is already used in the programs.

Display File Object (T8189DSP)

In this example a display file is used by the local program so that a user can enter requests that are to be sent to the remote program. The command used to create the display device file is:

```

CRTDSPF FILE(APPCLIB/T8189DSP) SRCFILE(QUSRTOOL/QATTDSS)
SRCMBR(TD8189) SRCMBR(TD8189)
TEXT('Display file for APPC examples')

```

The DDS source for the display device file T8189DSP is shown in Figure F-2.

```

A*****
A*
A*          DDS
A*          FOR THE DISPLAY FILE
A*          USED IN ITEM INQUIRY APPLICATIONS
A*
A*****
A*
A*          DSPSIZ(24 80 *DS3)
A*          INDARA
A*          CA03(99)
A*
A*****
A*          RECORD FORMATS
A*****
A*
A*          R PROMPT
A*
A*          PARTN      5A B 5 25
A*                   10 10'Part Number: '
A*          PARTD     25A 0 10 30
A*                   10 10'Part Description: '
A*          ERRORL    40A 0 12 10DSPATR(HI)
A*                   23 5'F3 = Exit'

```

Figure F-2. DDS Source for the Display Device File

Database File Object (T8189DB)

In this example the database file resides on the remote system and contains the part numbers and associated descriptions. The file is used to validate the part number received from the local program. The command used to create the database file (a physical file) is

```

CRTPF FILE(APPCLIB/T8189DB) SRCFILE(APPCLIB/QATTDSS) SRCMBR(TA8189)
TEXT('Database file for APPC examples')

```

The DDS source for the database file T8189DB is shown in Figure F-3.

```

A*****
A*
A*          DDS
A*          FOR THE DATABASE FILE
A*          USED IN ITEM INQUIRY APPLICATIONS
A*
A*****
A*
A*          UNIQUE
A*          R DBRCD
A*          ITEMNM      5
A*          ITEMND     25
A*          K ITEMNM

```

Figure F-3. DDS Source for the Database File

C/400 Local Program for Inquiry Applications (Example 1)

The following explains the structure of the C/400 local program that sends requests to the partner program for processing

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure F-4 on page F-5.

Note: On any type of error that is not expected (for example, an unexpected CPI Communications *return_code* on a call), the session is ended and the program ends.

1

The preprocessor include directive #include "QCC/H/CMC" replaces the directive with the contents of the AS/400 supplied CPI Communications pseudonym file CMC. Refer to "Using Pseudonyms When Writing Applications" on page 6-6 for further information on pseudonym files.

2

This section defines the display file (T8189DSP) structures used in the program. T8189DSP is the display file used to receive a

user's requests and to report the requested information received from the partner program based on the part number specified by the user. T8189DSP uses the file-level keyword, INDARA, which indicates that the file uses a separate indicator area.

- 3** The internal functions are prototyped so the C/400 compiler knows the type of value returned and the type of parameters passed, if any.
- 4** The `open_files`, `start_conversation`, and `get_cust_num` functions are called to open files used by the program, start a conversation with the partner program, and obtain the part number to be queried, respectively.
- 5** The program loops until either F3 is pressed from the work station, which sets the indicator in the separate indicator area of the display file, or an error occurs in the transaction with the partner program.
- 6** The part number is sent to the partner program using the `Send_Data` (CMSSEND) call. The CMSSEND call is issued with the following conversation characteristics (the conversation characteristics were set in `start_conversation`): a *send_type* of `CM_SEND_AND_PREP_TO_RECEIVE`; a *prepare_to_receive_type* of `CM_PREP_TO_RECEIVE_SYNC_LEVEL`; and a *sync_level* of `CM_CONFIRM`. Setting the conversation characteristics to these values flushes the data, changes the data flow direction from send to receive, and sends a confirmation request to the partner program. The partner program must now respond with a positive or negative response.
- 7** If the partner program responds with a positive response (*return_code* of `CM_OK`) to the confirmation request, a `Receive` (CMRCV) call is issued to receive the part description. However, if the partner program responds with a negative response (*return_code* of `CM_PROGRAM_ERROR_PURGING`) to the confirmation request, a `CMRCV` call is issued to receive the error message.
- 8** The `get_cust_num` function is called to display the information returned by the partner program and to obtain the next part number to be queried.
- 9** The `cleanup` function is called to perform end-of-program processing.
- 10** The `open_files` function opens the display file.
- 11** A separate indicator area is defined for the file T8189DSP. The variable `dsp_indic` is of the type `_SYSindara`, which is a 99-character array.
- 12** The separate indicator area for the display file T8189DSP is initialized.
- 13** The `start_conversation` function establishes a conversation with the partner program and sets various conversation characteristics.
- 14** The `Initialize_Conversation` (CMINIT) call is issued to initialize the conversation characteristics before the conversation is allocated.
Note: The *sym_dest_name* used is the side information object T8189CSI.
- 15** The `Set_TP_Name` (CMSTPN) call is issued so that the *TP_name* conversation characteristic is set to the remote program.
Note: The remote program that is to be started can be any of the remote programs in this appendix and in Appendix E, "Example Programs for Applications Using the ICF Application Program Interface" on page E-1.
- 16** The `Set_Sync_Level` (CMSSL) call is issued so that the *sync_level* conversation characteristic is set to `CM_CONFIRM`.
- 17** The `Allocate` (CMALLC) call is issued so that a conversation can be started using the *conversation_ID* previously assigned by the `CMINIT` call.
- 18** The `Set_Send_Type` (CMSST) call is issued so that the *send_type* conversation character-

istic is set to
CM_SEND_AND_PREP_TO_RECEIVE.

19

The `get_cust_num` function displays the requested information, and reads the next number. The part number field will be blank the first time a part number is read.

20

The `cleanup` function performs end-of-program processing.

21

If no error was detected, the `deallocate_type` is set to `CM_DEALLOCATE_FLUSH` by issuing the `Set_Deallocate_Type` (CMSDT) call, followed by a call to `Deallocate` (CMDEAL) to end the conversation normally.

Note: The CMSDT call must be performed because the default `deallocate_type` is `CM_DEALLOCATE_SYNC_LEVEL`, which would send a confirmation request to the partner program when the CMDEAL call is issued.

22

If an unexpected error was detected and the conversation is still active, the `deallocate_type` is set to `CM_DEALLOCATE_ABEND` by issuing the CMSDT call, followed by a call to `Deallocate` (CMDEAL) to end the conversation abnormally.

23

The display file is closed.

1

* * * * * P R O L O G * * * * *

```

Program name . . . . . : T8189CCS
Library name . . . . . : APPCLIB
Source file . . . . . : QATTSYSC
Library name . . . . . : QUSRTOOL
Source member name . . . . . : T8189CCS
Text Description . . . . . :
Compiler options . . . . . : *SOURCE *NOXREF *NOSHOWUSR *NOSHOWSYS *NOSHOWSKP *NOEXPMAC *NOAGR
: *NOPPNLY *NODEBUG *GEN *NOSECLVL *PRINT *LOGMSG
Language level options . . . . . : *EXTENDED
Source margins:
Left margin . . . . . : 1
Right margin . . . . . : 32767
Sequence columns:
Left Column . . . . . :
Right Column . . . . . :
Define name . . . . . :
Generation options . . . . . : *NOLIST *NOXREF *GEN *NOATR *NODUMP *NOOPTIMIZE *NOALWBND
: *NOANNO
Print file . . . . . : QSYSPT
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 . . . . . :
Last change . . . . . : 90/09/26 08:26:09
Source description . . . . . :
Compiler . . . . . : IBM C/400 Compiler

```

2

* * * * * S O U R C E * * * * *

```

Line STMT
1 /******
2 /* Program name.....: T8189CCS */
3 /* Program description.: CPI Communications local program */
4 /* Language.....: C/400 */
5 /*
6 /* This program invokes a program to handle part inquiry on */
7 /* the remote system. The Initialize_Conversation (CHINIT) */
8 /* call is issued using the sym_dest_name of 'T8189CSI'. */
9 /* The Allocate (CMALLC) call is issued, which results in */
10 /* the establishment of a conversation with the remote */
11 /* program. A display which prompts the user for the part */
12 /* number for which part information is requested is then */
13 /* displayed. When the user presses Enter, a Send_Data */
14 /* (CMSEND) call is issued (the data sent to the partner */
15 /* program is the part number). Note that the CMSEND call */
16 /* is issued with the following conversation */
17 /* characteristics: a send_type of */
18 /* CM_SEND_AND_PREP_TO_RECEIVE; a prepare_to_receive_type */
19 /* of CM_PREP_TO_RECEIVE_SYNC_LEVEL; and a sync_level of */
20 /* CM_CONFIRM. Setting the conversation characteristics to */
21 /* these values results in the flushing of the data, the */
22 /* changing of the data flow direction, and the sending of */
23 /* a confirmation request to the partner program. If the */
24 /* partner program responds with the Confirmed (CMCFMD) */
25 /* call to the confirmation request, the return_code */
26 /* parameter value on the CMSEND call will be set to CM_OK; */
27 /* the Receive (CMRCV) call is then issued to receive the */
28 /* part description. However, if the partner program */

```

Figure F-4 (Part 1 of 7). C/400 Inquiry Example – Local Program

```

29      /* responds with the Send_Error (CMSERR) call to the          */
30      /* confirmation request, the return_code parameter value on   */
31      /* the CMSEND call will be set to CM_PROGRAM_ERROR_PURGING;   */
32      /* a CMRCV call is issued to receive the error message.      */
33      /*                                                            */
34      /* The error message or part description (depending on       */
35      /* whether the part number was found) will be displayed on    */
36      /* the screen.                                              */
37      /*                                                            */
38      /* This program will continue to handle inquiries until the   */
39      /* user presses the F3=Exit key. When F3=Exit is pressed,    */
40      /* the Deallocate (CMDEAL) call is issued to end the        */
41      /* conversation (note that the deallocate_type conversation   */
42      /* characteristic is set to CM_DEALLOCATE_FLUSH), and        */
43      /* program processing ends.                                  */
44      /*                                                            */
45      /* NOTE 1: If an unexpected return_code value is received on  */
46      /*          any of the CPI Communications calls, the         */
47      /*          program will abnormally end the conversation (if  */
48      /*          it is still active), and program processing will  */
49      /*          end.                                              */
50      /*                                                            */
51      /* NOTE 2: On the CMRCV call, if the data_received          */
52      /*          parameter value does not indicate                 */

```

3

```

Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.....
53      /*          CM_COMPLETE_DATA_RECEIVED, or if the          */
54      /*          status_received parameter value does not indicate */
55      /*          CM_SEND_RECEIVED, the program will abnormally end */
56      /*          the conversation (if it is still active), and    */
57      /*          program processing will end.                    */
58      /*                                                            */
59      /* NOTE 3: This program can start ANY of the "remote"      */
60      /*          program examples in the APPC Programmer's      */
61      /*          Guide by changing the TP_name variable to the   */
62      /*          remote program that is to be started.         */
63      /*                                                            */
64      /******
65      |
66      /******
67      /* Retrieve various structures/utilities that are used in program. */
68      /******
69      #include <stdio.h>          /* Standard I/O header          */
70      #include <stdlib.h>        /* General utilities          */
71      #include <string.h>        /* String handling utilities  */
72      #include <stddef.h>        /* Standard definition        */
73      #include <xxfdbk.h>        /* Feedback area structures   */
74      #include <xxasio.h>        /* Indicator area structure   */
75      #include <recio.h>         /* Record i/o routines        */
76 1 #include "QCC/H/CMC"         /* CPI Communications pseudonyms */
77      |
78      /******
79      /* Define variables used with CPI Communications calls.    */
80      /******
81      CM_INT32 data_received;
82      CM_INT32 requested_length;
83      CM_INT32 received_length;
84      CM_INT32 request_to_send_received;
85      CM_INT32 return_code;
86      CM_INT32 send_length;
87      CM_INT32 send_type;
88      CM_INT32 status_received;
89      CM_INT32 deallocate_type;
90      CM_INT32 sync_level;
91      CM_INT32 TP_name_length;

```

Figure F-4 (Part 2 of 7). C/400 Inquiry Example – Local Program

```

92 |char  conversation_ID??(8??);
93 |char  sym_dest_name??(8??);
94 |char  TP_name??(8??);
95 |
96 |/*****
97 |/* Define the structures used for reads/writes from/to the display */
98 |/* file. */
99 |/*****
100 |2 | struct {
101 |   char partn??(5??);           /* part number           */
102 |   char partd??(25??);         /* part description      */
103 |   char error1??(40??);        /* error record          */
104 | } prompt_i_o = { "          ", "          ", "          ", "          " };
105 |
106 |
4
Line STMT
   *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
107 |/*****
108 |/* Define constants/flags used in program. */
109 |/*****
110 |#define ERROR 1                /* error during IO processing */
111 |#define NOERROR 0
112 |
113 |/*****
114 |/* Declare global variables/functions. */
115 |/*****
116 |3 | void cleanup(int);
117 |   void open_files(void);
118 |   void get_cust_num(void);
119 |   void start_conversation(void);
120 |
121 |   _RFILE *dspfptr;           /* Pointer to the display file */
122 |   XXIOFB_DSP_ICF_T *dsp_icf_fdbk; /* IO Feedback - display file */
123 |   _SYSindara dsp_indic;      /* indicator area for dsp */
124 |   char blank40??(40??) = "          ";
125 |
126 |
127 |/*****
128 |/* START OF PROGRAM */
129 |/*
130 |/* Files are opened, a conversation with the remote program is
131 |/* started, and the part inquiry screen is displayed. Inquiries
132 |/* are handled until the user presses the F3=Exit key, in which case
133 |/* the conversation will be ended and the program will end.
134 |/*****
135 |main()
136 |{
137 |4 |1 | open_files();
138 |   2 | start_conversation();
139 |
140 |   3 | get_cust_num();
141 |
142 |5 |4 | while (dsp_indic??(98??) != '1')
143 |   {
144 |
145 |   /*
146 |   /* The part number that the user has requested information
147 |   /* for is sent to the remote program using the CMSEND call.
148 |   /*
149 |   5 |   send_length = 5;
150 |   6 |   CMSEND(conversation_ID, prompt_i_o.partn, &send_length,
151 |           &request_to_send_received, &return_code);
152 |

```

Figure F-4 (Part 3 of 7). C/400 Inquiry Example – Local Program

```

153 | /*****
154 | /* The CMRCV call is issued to receive the response */
155 | /* from the remote program (the response can either be */
156 | /* an error message or the part description, depending */
157 | /* on whether the part was found or not). */
158 | /*****
159 | 7 7 | if (return_code == CM_OK)
160 |     {
5
Line STMT
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
161 | 8 |     requested_length = 25;
162 |     CMRCV(conversation_ID, prompt_i_o.pardt, &requested_length,
163 |           &data_received, &received_length, &status_received,
164 |           &request_to_send_received, &return_code);
165 |
166 | 10 |     strncpy(prompt_i_o.error1, blank40, 40);
167 |     }
168 |     else
169 | 11 |     if (return_code == CM_PROGRAM_ERROR_PURGING)
170 |     {
171 | 12 |         requested_length = 40;
172 |         CMRCV(conversation_ID, prompt_i_o.error1, &requested_length,
173 |               &data_received, &received_length, &status_received,
174 |               &request_to_send_received, &return_code);
175 |
176 | 14 |         strncpy(prompt_i_o.pardt, " ", 25);
177 |     }
178 |     else /* Unexpected return code. */
179 | 15 |     cleanup(ERROR);
180 |
181 |     if ((return_code != CM_OK) ||
182 |         (data_received != CM_COMPLETE_DATA_RECEIVED) ||
183 |         (status_received != CM_SEND_RECEIVED))
184 | 16 |     cleanup(ERROR);
185 |
186 | 8 18 |     get_cust_num();
187 |     } /* end of while */
188 |
189 | 9 19 | cleanup(NOERROR);
190 |
191 | } /* end of MAIN routine */
192 |
193 | /*****
194 | /*
195 | /* *****
196 | /* * INTERNAL FUNCTIONS *
197 | /* *****
198 | /*
199 | /*****
200 |
201 | /*****
202 | /* "OPEN_FILES" function
203 | /*
204 | /* This function opens the display file and sets the indicator area. */
205 | /*****
206 | 10 | void open_files()
207 |     {
208 |     if ((dspfptr= Ropen("T8189DSP", "a+ indicators=y riofb=y" ))
209 |         == NULL)
210 |     {
211 |         printf("Display file failed to open.\n");
212 |         exit(ERROR);
213 |     }
214 |     }

```

Figure F-4 (Part 4 of 7). C/400 Inquiry Example — Local Program

6

```

Line  STMT
      *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
215 11 4 | _Rindara(dspfp_ptr, dsp_indic);
216 |
217 12 5 | memset(dsp_indic, '0', 99);      /* Initialize indicator area.    */
218 | }                                /* end open_files...          */
219 |
220 | /*****
221 | /* "START_CONVERSATION" function
222 | /*
223 | /* This function establishes a conversation with the remote program, */
224 | /* and also sets various conversation characteristics.                */
225 | /*****
226 13 | void start_conversation()
227 | {
228 | /*****
229 | /* The CMINIT call is issued to initialize various
230 | /* conversation characteristics.
231 | /*****
232 | 1 | strncpy(sym_dest_name, "T8189CSI", 8);
233 14 2 | CMINIT(conversation_ID, sym_dest_name, &return_code);
234 | 3 | if (return_code != CM_OK)
235 | 4 |     cleanup(ERROR);
236 |
237 | /*****
238 | /* The Set_TP_Name (CMSTPN) call is issued so that the
239 | /* TP_name conversation characteristic is set to the remote program.*/
240 | /*****
241 | 5 | strncpy(TP_name, "T8189CCT", 8);
242 | 6 | TP_name_length = strlen(TP_name);
243 15 7 | CMSTPN(conversation_ID, TP_name, &TP_name_length, &return_code);
244 |
245 | /*****
246 | /* The Set_Sync_Level (CMSSSL) call is issued so that the
247 | /* sync_level conversation characteristic is set to CM_CONFIRM.
248 | /*****
249 | 8 | sync_level = CM_CONFIRM;
250 16 9 | CMSSSL(conversation_ID, sync_level, return_code);
251 |
252 | /*****
253 | /* The CMALLC call is issued so that a conversation can be started
254 | /* using the conversation_ID previously assigned by the CMINIT call.*/
255 | /*****
256 17 10 | CMALLC(conversation_ID, return_code)
257 | 11 | if (return_code != CM_OK)
258 | 12 |     cleanup(ERROR);
259 |
260 | /*****
261 | /* The Set_Send_Type (CMSST) call is issued so that the send_type
262 | /* conversation characteristic is set to
263 | /* CM_SEND_AND_PREP_TO_RECEIVE.
264 | /*****
265 | 13 | send_type = CM_SEND_AND_PREP_TO_RECEIVE;
266 18 14 | CMSST(conversation_ID, &send_type, &return_code);
267 |
268 | }                                /* end start_conversation...  */
7
Line  STMT
      *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
269 |
270 | /*****
271 | /* "GET_CUST_NUM" function
272 | /*
273 | /* Get a customer number from the display.
274 | /*****
275 19 | void get_cust_num()

```

Figure F-4 (Part 5 of 7). C/400 Inquiry Example – Local Program

```

276 |{
277 | 1 | _Rformat(dspfptr,"PROMPT  ");
278 | 2 | _Rwrite (dspfp, &prompt_i_o, sizeof(prompt_i_o));
279 | 3 | memset(dsp_indic, '0', 99);
280 | 4 | _Rreadn (dspfp, &prompt_i_o, sizeof(prompt_i_o), __DFT);
281 | } /* end get_cust_num... */
282 |
283 | /*****
284 | /* "CLEANUP" function. */
285 | /*
286 | /* The following code handles the end-of-program processing.
287 | /* This includes the ending of the conversation with
288 | /* the remote system (if conversation is active) by
289 | /* issuing a CMDEAL call with the deallocate_type set to either
290 | /* CM_DEALLOCATE_FLUSH (for non-error conditions),
291 | /* or CM_DEALLOCATE_ABEND (for error conditions).
292 | /*
293 | 20 | void cleanup(int endtype)
294 | |{
295 | 21 | 1 | if (endtype == NOERROR)
296 | | {
297 | 2 | deallocate_type = CM_DEALLOCATE_FLUSH;
298 | 3 | CMSDT(conversation_ID, &deallocate_type, &return_code);
299 | 4 | CMDEAL(conversation_ID, &return_code);
300 | | }
301 | | else
302 | 22 | | if ((return_code != CM_ALLOCATE_FAILURE_RETRY) &&
303 | | (return_code != CM_ALLOCATE_FAILURE_NO_RETRY) &&
304 | | (return_code != CM_DEALLOCATED_ABEND) &&
305 | | (return_code != CM_DEALLOCATED_NORMAL) &&
306 | | (return_code != CM_PRODUCT_SPECIFIC_ERROR) &&
307 | | (return_code != CM_RESOURCE_FAILURE_RETRY) &&
308 | 5 | | (return_code != CM_RESOURCE_FAILURE_NO_RETRY))
309 | | {
310 | 6 | deallocate_type = CM_DEALLOCATE_ABEND;
311 | 7 | CMSDT(conversation_ID, &deallocate_type, &return_code);
312 | 8 | CMDEAL(conversation_ID, &return_code);
313 | | }
314 | |
315 | 23 | 9 | _Rclose (dspfp);
316 | |
317 | 10 | exit(endtype);
318 | |} /* end cleanup... */
319 | |

```

```

* * * * * E N D   O F   S O U R C E   * * * * *
* * * * * I N C L U D E S   * * * * *

```

INCNO	Include Name	Last change	Actual Include Name
1	stdio.h	90/09/12 14:42:23	QCC/H/STDIO
2	stddef.h	90/09/12 14:42:22	QCC/H/STDDEF
3	errno.h	90/09/12 14:42:18	QCC/H/ERRNO
4	signal.h	90/09/12 14:42:22	QCC/H/SIGNAL
5	ctype.h	90/09/12 14:42:18	QCC/H/CTYPE
6	stdarg.h	90/09/12 14:42:22	QCC/H/STDARG
7	stdlib.h	90/09/12 14:42:24	QCC/H/STDLIB
8	string.h	90/09/12 14:42:24	QCC/H/STRING
9	stddef.h	90/09/12 14:42:22	QCC/H/STDDEF
10	xxfdbk.h	90/09/12 14:42:28	QCC/H/XXFDBK
11	xxasio.h	90/09/12 14:42:26	QCC/H/XXASIO
12	recio.h	90/09/12 14:42:21	QCC/H/RECIO
13	xxfdbk.h	90/09/12 14:42:28	QCC/H/XXFDBK
14	QCC/H/CMC	90/09/26 08:24:49	QCC/H/CMC

```

* * * * * E N D   O F   I N C L U D E S   * * * * *

```

Figure F-4 (Part 6 of 7). C/400 Inquiry Example – Local Program


```

          * * * * * 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 * * * * *
ROUTINE      BLOCK NUMBER  SCOPE  TYPE
<MAIN>         2          LOCAL  MAIN-PROGRAM
__memset       131         LOCAL  PROCEDURE
__strlen       145         LOCAL  PROCEDURE
__strncpy      151         LOCAL  PROCEDURE
QXXFORMAT      169         LOCAL  PROCEDURE
QXXINDARA      170         LOCAL  PROCEDURE
__reads        192         LOCAL  PROCEDURE
__rwrite       195         LOCAL  PROCEDURE
__rfmt         202         LOCAL  PROCEDURE
cleanup        248         ENTRY  PROCEDURE
open_files     249         ENTRY  PROCEDURE
get_cust_num   250         ENTRY  PROCEDURE
start_conversation
                251         ENTRY  PROCEDURE
main           256         ENTRY  PROCEDURE

```

Figure F-4 (Part 7 of 7). C/400 Inquiry Example – Local Program

C/400 Remote Program for Inquiry Applications (Example 1)

The following explains the structure of the C/400 remote program that handles requests sent by the partner program.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure F-5 on page F-13.

Note: On any type of error that is not expected (for example, an unexpected CPI Communications *return_code* on a call), the session is ended and the program ends.

- 1** The preprocessor include directive `#include "QCC/H/CMC"` causes the preprocessor to replace the directive with the contents of the AS/400-supplied CPI Communications pseudonym file CMC.
- 2** This structure defines the database file (T8189DB) structures used in the program. T8189DB is the database file used to read the customer records.
- 3** The internal functions are prototyped so the C/400 compiler knows the type of value returned and the type of parameters passed, if any.
- 4** The `open_files` and `start_conversation` functions are called to open files used by the program and to start a conversation with the partner program, respectively.
- 5** The program loops until there are no more requests to process or until an error occurs in the transaction with the partner program.
- 6** A search of the database file is performed using the part number received from the partner program as the key.
- 7** If the part number is found in the database file, the Confirmed (CMCFMD) call is issued. As a result, a positive response to the received confirmation request is sent to the partner program.
- 8** If the part number is not found in the database file, the `Send_Error` (CMSERR) call is issued. As a result, a negative response to the received confirmation request is sent to the partner program.
- 9** The `Send_Data` (CMSSEND) call is issued. The data sent (set previously) is either an error message (if the part was not found) or the part description (if the part is found). The CMSSEND call is issued with the following conversation characteristics (the conversation characteristics were set in `start_conversation`): a *send_type* of `CM_SEND_AND_PREP_TO_RECEIVE` and a *prepare_to_receive_type* of `CM_PREP_TO_RECEIVE_FLUSH`. Setting the conversation characteristics to these values

flushes the data and changes the data flow direction.

10 The `cleanup` function is called to perform end-of-program processing.

11 The `open_files` function opens the database file.

12 The `start_conversation` function establishes a conversation with the partner program and sets various conversation characteristics.

13 The `Accept_Conversation` (CMACCP) call is issued so that a conversation can be started with the partner program.

14 The `Set_Send_Type` (CMSST) call is issued so that the `send_type` conversation characteristic is set to `CM_SEND_AND_PREP_TO_RECEIVE`.

15 The `Set_Prepare_To_Receive_Type` (CMSPTR) call is issued so that the `prepare_to_receive_type` conversation characteristic is set to `CM_PREP_TO_RECEIVE_FLUSH`.

16 The `get_cust_num` function waits for a request from the partner program by issuing the `Receive` (CMRCV) call.

Note: A transaction is processed if all the data is received with a turnaround indication, and the partner program requested confirmation. This is indicated by the following:

- The value of the `data_received` variable on the CMRCV call is `CM_COMPLETE_DATA_RECEIVED`
- The value of the `status_received` variable on the CMRCV call is `CM_CONFIRM_SEND_RECEIVED`

17 The `cleanup` function performs end-of-program processing.

18 If the conversation is still active, it is assumed that an error was detected. The `deallocate_type` is set to `CM_DEALLOCATE_ABEND` by issuing the `Set_Deallocate_Type` (CMSDT) call, followed by a call to `Deallocate` (CMDEAL) to end the conversation abnormally.

19 The database file is closed.

```

          * * * * * P R O L O G * * * * *
Program name . . . . . : T8189CCT
Library name . . . . . : APPCLIB
Source file . . . . . : QATTSYSC
Library name . . . . . : QUSRTOOL
Source member name . . . . . : T8189CCT
Text Description . . . . . :
Compiler options . . . . . : *SOURCE *NOXREF *NOSHOWUSR *NOSHOWSYS *NOSHOWSKP *NOEXPMAC *NOAGR
                          : *NOPPONLY *NODEBUG *GEN *NOSECLVL *PRINT *LOGMSG
Language level options . . . . . : *EXTENDED
Source margins:
Left margin . . . . . : 1
Right margin . . . . . : 32767
Sequence columns:
Left Column . . . . . :
Right Column . . . . . :
Define name . . . . . :
Generation options . . . . . : *NOLIST *NOXREF *GEN *NOATR *NODUMP *NOOPTIMIZE *NOALWBND
                          : *NOANNO
Print file . . . . . : QSYSPT
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 . . . . . :
Last change . . . . . : 90/09/26 08:26:11
Source description . . . . . :
Compiler . . . . . : IBM C/400 Compiler

```

* * * * * S O U R C E * * * * *

```

Line STMT
1  /*.....1.....2.....3.....4.....5.....6.....7.....8.....9.....
2  /******
3  /* Program name.....: T8189CCT */
4  /* Program description.: CPI Communications remote program */
5  /* Language.....: C/400 */
6  /*
7  /* This program accepts the incoming conversation by
8  /* issuing an Accept_Conversation (CMACCP) call. It then
9  /* issues a Receive (CMRCV) call to receive the part number
10 /* from the remote system. When the CMRCV call completes,
11 /* the status_received value will be CM_CONFIRM_SEND. The
12 /* database file T8189DB is searched for the received part
13 /* number. If the part number is found, the Confirmed
14 /* (CMCFMD) call is issued, followed by a Send_Data
15 /* (CMSEND) call (the data sent is the part description
16 /* corresponding to the part number retrieved from the
17 /* database file). However, if the part number is not
18 /* found, the Send_Error (CMSERR) call is issued, followed
19 /* by a CMSEND call (the data sent is a message describing
20 /* the error). The CMSEND call sending either the part
21 /* description or the error message is issued with a
22 /* send_type conversation characteristic of
23 /* CM_SEND_AND_PREP_TO_RECEIVE and a
24 /* prepare_to_receive_type conversation characteristic of
25 /* CM_PREP_TO_RECEIVE_FLUSH. Setting the conversation
26 /* characteristics to these values results in the flushing
27 /* of the data, and the changing of the data flow
28 /* direction. The partner program can send more inquiries.
29 /*
30 /* This program will continue to handle inquiries from the
31 /* partner program until a return_code that is not CM_OK
32 /* is received. Then the program ends.

```

Figure F-5 (Part 1 of 6). C/400 Inquiry Example – Remote Program

```

32      /*
33      /* NOTE 1: If an unexpected return_code value is received on
34      /*   any of the CPI Communications calls, the
35      /*   program will abnormally end the conversation
36      /*   with a deallocate_type of CM_DEALLOCATED_ABEND,
37      /*   and program processing will end.
38      /*
39      /* NOTE 2: On the CMRCV call, if the data_received
40      /*   parameter value does not indicate
41      /*   CM_COMPLETE_DATA_RECEIVED, or if the
42      /*   status_received parameter value does not indicate
43      /*   CM_CONFIRM_SEND_RECEIVED, the program will
44      /*   abnormally end the conversation with a
45      /*   deallocate_type of CM_DEALLOCATED_ABEND,
46      /*   and program processing will end.
47      /*
48      /* NOTE 3: This program can be started by ANY of the
49      /*   "local" program examples in the APPC Programmer's
50      /*   Guide.
51      /*
52      /******
Line STMT      *...+...1....+...2....+...3....+...4....+...5....+...6....+...7....+...8....+...9.....
53      |
54      /******
55      /* Retrieve various structures/utilities that are used in program.
56      /******
57      #include <stdio.h>          /* Standard I/O header
58      #include <stdlib.h>       /* General utilities
59      #include <string.h>       /* String handling utilities
60      #include <stddef.h>       /* Standard definition
61      #include <xxfdbk.h>       /* Feedback area structures
62      #include <xxasio.h>       /* Indicator area structure
63      #include <recio.h>        /* record i/o routines
64      1 #include "QCC/H/CMC"     /* CPI Communications pseudonyms
65      |
66      /******
67      /* Define the structure used for reads from the database file.
68      /******
69      2 | struct
70      |   | char partn??(5??);
71      |   | char partd??(25??);
72      | } part_rec;
73      |
74      /******
75      /* Define variables used with CPI Communications calls.
76      /******
77      CM_INT32 data_received;
78      CM_INT32 prep_to_receive_type;
79      CM_INT32 requested_length;
80      CM_INT32 received_length;
81      CM_INT32 request_to_send_received;
82      CM_INT32 return_code;
83      CM_INT32 send_length;
84      CM_INT32 send_type;
85      CM_INT32 status_received;
86      CM_INT32 deallocate_type;
87      |char conversation_ID??(8??);
88      |char buffer??(40??);
89      |
90      /******
91      /* Define constants/flags used in program.
92      /******
93      #define ERROR 1           /* error during I/O processing
94      #define NOERROR 0
95      #define MORE_REQUESTS 0   /* More request indicator
96      #define NO_REQUESTS 1

```

Figure F-5 (Part 2 of 6). C/400 Inquiry Example – Remote Program

```

97 |
98 | /*****
99 | /* Declare global variables/functions. */
100 | /*****
101 | 3 | int get_cust_num(void);
102 | void cleanup(void);
103 | void open_files(void);
104 | void start_conversation(void);
105 |
106 | char part_not_found??(40??) =
4
Line STMT
107 | *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
108 | | "THE REQUESTED PART WAS NOT FOUND ";
109 | | _RFILE *dbfptr; /* Pointer to database file. */
110 | | _RIOFB_T *db_fdbk; /* IO Feedback - database file */
111 | |
112 | | size_t size; /* "size_t" is a synonym for the */
113 | | /* type of the value returned by */
114 | | /* the "sizeof" operator. */
115 | |
116 | |
117 | | /*****
118 | | /* START OF PROGRAM */
119 | | /*
120 | | /* Files are opened, a conversation with the remote program is
121 | | /* started, and the part inquiry processing starts. Inquiries
122 | | /* are handled until a CM_DEALLOCATED_NORMAL is received.
123 | | /*****
124 | | main()
125 | | {
126 | | 4 | 1 | open_files();
127 | | 2 | start_conversation();
128 | |
129 | | 5 | 3 | while (get_cust_num() != NO_REQUESTS)
130 | | | {
131 | | | /*****
132 | | | /* A search of the database file is done using the part */
133 | | | /* number as the key. */
134 | | | /*****
135 | | | 4 | strncpy (part_rec.partd," ",25);
136 | | | 6 | db_fdbk = _Rreadk(dbfptr, &part_rec, sizeof(part_rec),
137 | | | 5 | _KEY_EQ, &part_rec.partn, sizeof(part_rec.partn));
138 | | |
139 | | | /*****
140 | | | /* If the part number is found, the CMCFMD call is */
141 | | | /* issued; otherwise, the CMSERR call is issued. */
142 | | | /*****
143 | | | 7 | 6 | if (db_fdbk -> num_bytes > 0) /* if record was found */
144 | | | | {
145 | | | | CMCFMD(conversation_ID, &return_code);
146 | | | | 8 | if (return_code != CM_OK)
147 | | | | 9 | cleanup();
148 | | | |
149 | | | | 10 | strncpy(buffer,part_rec.partd,25);
150 | | | | 11 | send_length = 25;
151 | | | | }
152 | | | | else /* part description not found */
153 | | | | {
154 | | | | 8 | 12 | CMSERR(conversation_ID, &request_to_send_received, &return_code);
155 | | | | 13 | if (return_code != CM_OK)
156 | | | | 14 | cleanup();
157 | | | |
158 | | | | 15 | strncpy(buffer, part_not_found, 40);
159 | | | | 16 | send_length = 40;
160 | | | | }

```

Figure F-5 (Part 3 of 6). C/400 Inquiry Example – Remote Program

```

Line  STMT
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
161  |
162  | /*****
163  | /* The CMSEND call is issued. The data sent (set previously) */
164  | /* is either an error message (if the part was not found) */
165  | /* or the part description (if the part is found). */
166  | /*****
167  9  | CMSEND(conversation_ID, buffer, &send_length,
168  17 |         &request_to_send_received, &return_code);
169  18 | if (return_code != CM_OK)
170  19 |     cleanup();
171  |     } /* end WHILE */
172  |
173  10 20 | cleanup();
174  |     } /* end of main routine */
175  |
176  | /*****
177  | /*
178  | /* *****
179  | /* * INTERNAL FUNCTIONS *
180  | /* *****
181  | /*
182  | /*****
183  |
184  | /*****
185  | /* "OPEN_FILES" function
186  | /*
187  | /* This function opens the database file.
188  | /*****
189  11  | void open_files()
190  | {
191  1  | if ((dbfptr= _Ropen("T8189DB", "rr riofb=n")) == NULL)
192  |     {
193  2  |     printf("Database file failed to open.\n");
194  3  |     exit(ERROR);
195  |     }
196  | } /* end open_files... */
197  |
198  | /*****
199  | /* "START_CONVERSATION" function
200  | /*
201  | /* This function establishes a conversation with the remote system,
202  | /* and also sets various conversation characteristics.
203  | /*****
204  12  | void start_conversation()
205  | {
206  |
207  | /*****
208  | /* The CMACCP call is issued so that a conversation can be
209  | /* started with the partner program.
210  | /*****
211  13 1 | CMACCP(conversation_ID, &return_code);
212  2  | if (return_code != CM_OK)
213  3  |     cleanup();
214  |
6
Line  STMT
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
215  | /*****
216  | /* The Set_Send_Type (CMSST) call is issued so that the
217  | /* send_type conversation characteristic is set to
218  | /* CM_SEND_AND_PREP_TO_RECEIVE.
219  | /*****

```

Figure F-5 (Part 4 of 6). C/400 Inquiry Example – Remote Program

```

220 4 | send_type = CM_SEND_AND_PREP_TO_RECEIVE;
221 14 5 | CMSST(conversation_ID, &send_type, &return_code);
222 |
223 | /*****
224 | /* The Set_Prepare_To_Receive_Type (CMSPTR) call is issued */
225 | /* so that the prepare_to_receive_type conversation */
226 | /* characteristic is set to CM_PREP_TO_RECEIVE_FLUSH. */
227 | *****/
228 6 | prep_to_receive_type = CM_PREP_TO_RECEIVE_FLUSH;
229 15 7 | CMSPTR(conversation_ID, &prep_to_receive_type, &return_code);
230 |
231 | } /* end start_conversation... */
232 |
233 | /*****
234 | /* "GET_CUST_NUM" function */
235 | /*
236 | /* This subroutine waits for incoming data from the partner */
237 | /* program by issuing the read operation. */
238 | *****/
239 16 | get_cust_num()
240 | {
241 1 | requested_length = 5;
242 | CMRCV(conversation_ID, part_rec.partn, &requested_length,
243 | &data_received, &received_length, &status_received,
244 2 | &request_to_send_received, &return_code);
245 |
246 3 | if (return_code == CM_OK)
247 | {
248 | if ((data_received == CM_COMPLETE_DATA_RECEIVED) &&
249 4 | (status_received == CM_CONFIRM_SEND_RECEIVED))
250 5 | return(MORE_REQUESTS);
251 | else
252 6 | return(NO_REQUESTS);
253 | }
254 | else
255 7 | return(NO_REQUESTS);
256 | } /* end get_cust_num... */
257 |
258 | /*****
259 | /* "CLEANUP" function. */
260 | /*
261 | /* The following code handles the end-of-program processing. */
262 | /* This includes the ending of the conversation with */
263 | /* the remote system (if conversation is active), and the */
264 | /* closing of opened files. */
265 | *****/
266 17 | void cleanup()
267 | {
268 18 | if ((return_code != CM_ALLOCATE_FAILURE_RETRY) &&
Line STMT *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9.....
269 | (return_code != CM_ALLOCATE_FAILURE_NO_RETRY) &&
270 | (return_code != CM_DEALLOCATED_ABEND) &&
271 | (return_code != CM_DEALLOCATED_NORMAL) &&
272 | (return_code != CM_PRODUCT_SPECIFIC_ERROR) &&
273 | (return_code != CM_RESOURCE_FAILURE_RETRY) &&
274 1 | (return_code != CM_RESOURCE_FAILURE_NO_RETRY))
275 | {
276 2 | deallocate_type = CM_DEALLOCATE_ABEND;
277 3 | CMSDT(conversation_ID, &deallocate_type, &return_code);
278 4 | CMDEAL(conversation_ID, &return_code);
279 | }
280 |

```

Figure F-5 (Part 5 of 6). C/400 Inquiry Example – Remote Program

```

281 19 5 | _Rclose(dbfptr);
282      |
283      6 | exit(0);
284      |}
285      |
                                     /* end cleanup...          */
                                     * * * * * END OF SOURCE * * * * *
                                     * * * * * INCLUDES * * * * *
INCN0  Include Name                Last change          Actual Include Name
  1  stdio.h                        90/09/12 14:42:23   QCC/H/STDIO
  2  stddef.h                       90/09/12 14:42:22   QCC/H/STDDEF
  3  errno.h                        90/09/12 14:42:18   QCC/H/ERRNO
  4  signal.h                       90/09/12 14:42:22   QCC/H/SIGNAL
  5  ctype.h                        90/09/12 14:42:18   QCC/H/CTYPE
  6  stdarg.h                       90/09/12 14:42:22   QCC/H/STDARG
  7  stdlib.h                       90/09/12 14:42:24   QCC/H/STDLIB
  8  string.h                       90/09/12 14:42:24   QCC/H/STRING
  9  stddef.h                       90/09/12 14:42:22   QCC/H/STDDEF
 10  xxfdbk.h                       90/09/12 14:42:28   QCC/H/XXFDBK
 11  xxasio.h                       90/09/12 14:42:26   QCC/H/XXASIO
 12  recio.h                        90/09/12 14:42:21   QCC/H/RECIO
 13  xxfdbk.h                       90/09/12 14:42:28   QCC/H/XXFDBK
 14  QCC/H/CMC                      90/09/26 08:24:49   QCC/H/CMC
                                     * * * * * END OF INCLUDES * * * * *
                                     * * * * * MESSAGE SUMMARY * * * * *
Total  Info(0-4)  Warning(5-19)  Error(20-29)  Severe(30-39)  Terminal(40-99)
      0          0          0          0          0          0
                                     * * * * * END OF MESSAGE SUMMARY * * * * *
ROUTINE      BLOCK NUMBER  SCOPE  TYPE
<MAIN>             2      LOCAL  MAIN-PROGRAM
__strncpy        151      LOCAL  PROCEDURE
__readk          194      LOCAL  PROCEDURE
get_cust_num     246      ENTRY  PROCEDURE
cleanup          247      ENTRY  PROCEDURE
open_files       248      ENTRY  PROCEDURE
start_conversation
                 249      ENTRY  PROCEDURE
main             254      ENTRY  PROCEDURE

```

Figure F-5 (Part 6 of 6). C/400 Inquiry Example – Remote Program

COBOL/400 Local Program for Inquiry Applications (Example 2)

The following explains the structure of the COBOL/400 local program that sends requests to the partner program for processing.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure F-6 on page F-20.

Note: On any type of error that is not expected (for example, an unexpected CPI Communications *return_code* on a call), the session is ended and the program ends.

1

The files used in the program are described in the file control section.

T8189DSP is the name of the display device file that is used to request an entry from the work station and to display the results of the inquiry. T8189DSP uses the file-level

keyword, INDARA, which indicates that the file uses a separate indicator area.

2

The COPY statement COPY CMCOBOL IN QLBL-QILBINC places the contents of the AS/400-supplied CPI Communications pseudonym file CMCOBOL in the program. Refer to “Using Pseudonyms When Writing Applications” on page 6-6 for further information on pseudonym files.

3

The OPEN-FILES, START-CONVERSATION, and GET-CUST-NUM routines are called to open files used by the program, start a conversation with the partner program, and obtain the part number to be queried, respectively.

4

The program loops until either F3 is pressed from the work station, which sets the indicator in the separate indicator area of the display file, or an error occurs in the transaction with the partner program.

5 The CLEAN-UP routine is called to perform end-of-program processing.

6 The OPEN-FILES routine opens the display file.

7 The separate indicator area for the display file T8189DSP is initialized.

8 The START-CONVERSATION routine establishes a conversation with the partner program.

9 The Initialize_Conversation (CMINIT) call is issued to initialize various conversation characteristics before the conversation is allocated.

Note: The *sym_dest_name* used is the side information object T8189CSI.

10 The Set_TP_Name (CMSTPN) call is issued so that the *TP_name* conversation characteristic is set to the remote program.

Note: The remote program that is to be started can be any of the remote programs in this appendix and in Appendix E, "Example Programs for Applications Using the ICF Application Program Interface" on page E-1.

11 The Set_Sync_Level (CMSSL) call is issued so that the *sync_level* conversation characteristic is set to CM_CONFIRM.

12 The Allocate (CMALLC) call is issued so that a conversation can be started using the *conversation_ID* previously assigned by the CMINIT call.

13 The Set_Send_Type (CMSST) call is issued so that the *send_type* conversation characteristic is set to CM_SEND_AND_PREP_TO_RECEIVE.

14 The HANDLE-INQUIRY routine contains the body of the loop that sends requests to the partner program.

15 The part number is sent to the partner program using the Send_Data (CMSEND) call.

The CMSEND call is issued with the following conversation characteristics (the conversation characteristics were set in START-CONVERSATION): a *send_type* of CM_SEND_AND_PREP_TO_RECEIVE; a *prepare_to_receive_type* of CM_PREP_TO_RECEIVE_SYNC_LEVEL; and a *sync_level* of CM_CONFIRM. Setting the conversation characteristics to these values flushes the data, changes the data flow direction from send to receive, and sends a confirmation request to the partner program. The partner program must now respond with a positive or negative response.

16 If the partner program responds with a positive response (*return_code* of CM_OK) to the confirmation request, a Receive (CMRCV) call is issued to receive the part description. However, if the partner program responds with a negative response (*return_code* of CM_PROGRAM_ERROR_PURGING) to the confirmation request, a CMRCV call is issued to receive the error message.

17 The GET-CUST-NUM routine is called to display the information returned by the partner program and to obtain the next part number to be queried.

18 The GET-CUST-NUM routine displays the requested information, and reads the next number. The part number field will be blank the first time a part number is read.

19 The CLEAN-UP routine performs end-of-program processing.

20 If the conversation is active, the Deallocate (CMDEAL) call is issued to end the conversation.

Note: The *deallocate_type* (set previously in the program) is either set to CM_DEALLOCATE_FLUSH if no error was detected, or CM_DEALLOCATE_ABEND if an error was detected.

21 The display file is closed.

```

Program . . . . . : T8189CLS
Library . . . . . : APPCLIB
Source file . . . . . : QATTCBL
Library . . . . . : QUSRT00L
Source member . . . . . : T8189CLS      09/26/90 08:27:00
Generation severity level . . . . . : 29
Text 'description' . . . . . : *BLANK
Source listing options . . . . . : *NONE
Generation options . . . . . : *NONE
Message limit:
  Number of messages . . . . . : *NOMAX
  Message limit severity . . . . . : 29
Print file . . . . . : QSYSVRT
Library . . . . . : *LIBL
FIPS flagging . . . . . : *NOFIPS *NOSEG *NODEB *NOOBSOLETE
SAA flagging . . . . . : *NOFLAG
Flagging severity . . . . . : 0
Replace program . . . . . : *YES
Target release . . . . . : *CURRENT
User profile . . . . . : *USER
Authority . . . . . : *LIBCRTAUT
Compiler . . . . . : IBM AS/400 COBOL/400
STMT SEQNBR -A 1 B.....2.....3.....4.....5.....6.....7..IDENTFCN S COPYNAME  CHG DATE
1 000010 IDENTIFICATION DIVISION.
000020
2 000030 PROGRAM-ID. T8191CLS.
000040*****
000050* Program name.....: T8189CLS *
000060* Program description.: CPI Communications local program *
000070* Language.....: COBOL/400 *
000080* *
000090* This program invokes a program to handle part inquiry on *
000100* the remote system. The Initialize_Conversation (CMINIT) *
000110* call is issued using the sym_dest_name of 'T8189CSI'. *
000120* The Allocate (CMALLC) call is issued, which results in *
000130* the establishment of a conversation with the remote *
000140* program. A display which prompts the user for the part *
000150* number for which part information is requested is then *
000160* displayed. When the user presses Enter, a Send_Data *
000170* (CMSEND) call is issued (the data sent to the partner *
000180* program is the part number). Note that the CMSEND call *
000190* is issued with the following conversation *
000200* characteristics: a send_type of *
000210* CM_SEND_AND_PREP_TO_RECEIVE; a prepare_to_receive_type *
000220* of CM_PREP_TO_RECEIVE_SYNC_LEVEL; and a sync_level of *
000230* CM_CONFIRM. Setting the conversation characteristics to *
000240* these values results in the flushing of the data, the *
000250* changing of the data flow direction, and the sending of *
000260* a confirmation request to the partner program. If the *
000270* partner program responds with the Confirmed (CMCFMD) *
000280* call to the confirmation request, the return_code *
000290* parameter value on the CMSEND call will be set to CM_OK; *
000300* the Receive (CMRCV) call is then issued to receive the *
000310* part description. However, if the partner program *
000320* responds with the Send_Error (CMSERR) call to the *
000330* confirmation request, the return_code parameter value on *
000340* the CMSEND call will be set to CM_PROGRAM_ERROR_PURGING; *
000350* a CMRCV call is issued to receive the error message. *
000360* *
000370* The error message or part description (depending on *
000380* whether the part number was found) will be displayed on *
000390* the screen. *
000400* *
000410* This program will continue to handle inquiries until the *
000420* user presses the F3=Exit key. When F3=Exit is pressed, *
000430* the Deallocate (CMDEAL) call is issued to end the *
000440* conversation (note that the deallocate_type conversation *
000450* characteristic is set to CM_DEALLOCATE_FLUSH), and *
000460* program processing ends. *
000470* *

```

Figure F-6 (Part 1 of 8). COBOL/400 Inquiry Example – Local Program

```

000480* NOTE 1: If an unexpected return_code value is received on *
000490*     any of the CPI Communications calls, the *
000500*     program will abnormally end the conversation (if *
000510*     it is still active), and program processing will *
000520*     end. *
000530* *
000540* NOTE 2: On the CMRCV call, if the data_received *
000550*     parameter value does not indicate *
STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME  CHG DATE
000560*     CM_COMPLETE_DATA_RECEIVED, or if the *
000570*     status_received parameter value does not indicate *
000580*     CM_SEND_RECEIVED, the program will abnormally end *
000590*     the conversation (if it is still active), and *
000600*     program processing will end. *
000610* *
000620* NOTE 3: This program can start ANY of the "remote" *
000630*     program examples in the APPC Programmer's *
000640*     Guide by changing the TP-NAME variable to the *
000650*     remote program that is to be started. *
000660* *
000670*****
000680
3 000690 ENVIRONMENT DIVISION.
000700
4 000710 CONFIGURATION SECTION.
000720
5 000730 SOURCE-COMPUTER. IBM-AS400.
6 000740 OBJECT-COMPUTER. IBM-AS400.
7 000750 SPECIAL-NAMES.
000760
8 000770 INPUT-OUTPUT SECTION.
000780
9 000790 FILE-CONTROL.
000800
1 10 000810     SELECT T8189DSP ASSIGN TO WORKSTATION-T8189DSP
11 000820         ORGANIZATION IS TRANSACTION
12 000830         CONTROL-AREA IS DISPLAY-FEEDBACK
13 000840         FILE STATUS IS STATUS-DSP.
000850
14 000860 DATA DIVISION.
000870
15 000880 FILE SECTION.
000890
000900*****
000910* FILE DESCRIPTION FOR THE DISPLAY FILE FOR THIS PROGRAM. *
000920*****
000930
16 000940 FD T8189DSP
17 000950     LABEL RECORDS ARE STANDARD.
18 000960 01 DSPREC.
19 000970     COPY DDS-ALL-FORMATS-I-O OF T8189DSP.
20 +000001     05 T8189DSP-RECORD PIC X(70). <-ALL-FMTS
+000002* INPUT FORMAT:PROMPT FROM FILE T8189DSP OF LIBRARY APPCLIB <-ALL-FMTS
+000003* <-ALL-FMTS
21 +000004     05 PROMPT-I REDEFINES T8189DSP-RECORD. <-ALL-FMTS
22 +000005     06 PARTN PIC X(5). <-ALL-FMTS
+000006* OUTPUT FORMAT:PROMPT FROM FILE T8189DSP OF LIBRARY APPCLIB <-ALL-FMTS
+000007* <-ALL-FMTS
23 +000008     05 PROMPT-O REDEFINES T8189DSP-RECORD. <-ALL-FMTS
24 +000009     06 PARTN PIC X(5). <-ALL-FMTS
25 +000010     06 PARTD PIC X(25). <-ALL-FMTS
26 +000011     06 ERRORL PIC X(40). <-ALL-FMTS
000980
27 000990 WORKING-STORAGE SECTION.
STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME  CHG DATE
001000
28 001010 77 STATUS-DSP PIC XX.
001020
29 001030 01 CONVERSATION-STATUS PIC 9(9) COMP-4.
30 001040 88 CONVERSATION-INITIALIZED VALUE 1.

```

Figure F-6 (Part 2 of 8). COBOL/400 Inquiry Example – Local Program

```

001050
31 001060 01 TR-CTL-AREA.
32 001070 05 FILLER PIC X(2).
33 001080 05 PGM-DEV-NME PIC X(10).
34 001090 05 RCD-FMT-NME PIC X(10).
001100
35 001110 01 DSPF-INDIC-AREA.
36 001120 05 CMD3 PIC 1 INDIC 99.
37 001130 88 CMD3-ON VALUE B"1".
38 001140 88 CMD3-OFF VALUE B"0".
001150
001160
39 001170 01 DISPLAY-FEEDBACK.
40 001180 05 CMD-KEY PIC XX.
41 001190 05 FILLER PIC X(10).
42 001200 05 RCD-FMT PIC X(10).
001210
001220*****
001230* Use the CPI Communications supplied pseudonyms. *
001240*****
001250
2 43 001260 COPY CMCOBOL IN QLBL-QILBINC.
+000010*COPY CMCOBOL CMCOBOL
+000020***** CMCOBOL
+000030* NOTE: BUFFER MUST BE DEFINED IN WORKING STORAGE * CMCOBOL
+000040***** CMCOBOL
+000050* CMCOBOL
44 +000060 01 CONVERSATION-ID PIC X(8). CMCOBOL
+000070* CMCOBOL
45 +000080 01 CONVERSATION-TYPE PIC 9(9) COMP-4. CMCOBOL
46 +000090 88 CM-BASIC-CONVERSATION VALUE 0. CMCOBOL
47 +000100 88 CM-MAPPED-CONVERSATION VALUE 1. CMCOBOL
+000110* CMCOBOL
48 +000120 01 CM-RETCODE PIC 9(9) COMP-4. CMCOBOL
+000130* ==> RETURN-CODE IS A RESERVED WORD IN SOME <==== CMCOBOL
+000140* ==> VERSIONS OF COBOL <==== CMCOBOL
+000150* CMCOBOL
49 +000160 88 CM-OK VALUE 0. CMCOBOL
50 +000170 88 CM-ALLOCATE-FAILURE-NO-RETRY VALUE 1. CMCOBOL
51 +000180 88 CM-ALLOCATE-FAILURE-RETRY VALUE 2. CMCOBOL
52 +000190 88 CM-CONVERSATION-TYPE-MISMATCH VALUE 3. CMCOBOL
53 +000200 88 CM-PIP-NOT-SPECIFIED-CORRECTLY VALUE 5. CMCOBOL
54 +000210 88 CM-SECURITY-NOT-VALID VALUE 6. CMCOBOL
55 +000220 88 CM-SYNC-LVL-NOT-SUPPORTED-LU VALUE 7. CMCOBOL
56 +000230 88 CM-SYNC-LVL-NOT-SUPPORTED-PGM VALUE 8. CMCOBOL
57 +000240 88 CM-TPN-NOT-RECOGNIZED VALUE 9. CMCOBOL
58 +000250 88 CM-TP-NOT-AVAILABLE-NO-RETRY VALUE 10. CMCOBOL
59 +000260 88 CM-TP-NOT-AVAILABLE-RETRY VALUE 11. CMCOBOL
60 +000270 88 CM-DEALLOCATED-ABEND VALUE 17. CMCOBOL
61 +000280 88 CM-DEALLOCATED-NORMAL VALUE 18. CMCOBOL
STMT SEQNBR -A 1 B...2...3...4...5...6...7..IDENTFCN S COPYNAME CHG DATE
62 +000290 88 CM-PARAMETER-ERROR VALUE 19. CMCOBOL
63 +000300 88 CM-PRODUCT-SPECIFIC-ERROR VALUE 20. CMCOBOL
64 +000310 88 CM-PROGRAM-ERROR-NO-TRUNC VALUE 21. CMCOBOL
65 +000320 88 CM-PROGRAM-ERROR-PURGING VALUE 22. CMCOBOL
66 +000330 88 CM-PROGRAM-ERROR-TRUNC VALUE 23. CMCOBOL
67 +000340 88 CM-PROGRAM-PARAMETER-CHECK VALUE 24. CMCOBOL
68 +000350 88 CM-PROGRAM-STATE-CHECK VALUE 25. CMCOBOL
69 +000360 88 CM-RESOURCE-FAILURE-NO-RETRY VALUE 26. CMCOBOL
70 +000370 88 CM-RESOURCE-FAILURE-RETRY VALUE 27. CMCOBOL
71 +000380 88 CM-UNSUCCESSFUL VALUE 28. CMCOBOL
72 +000390 88 CM-DEALLOCATED-ABEND-SVC VALUE 30. CMCOBOL
73 +000400 88 CM-DEALLOCATED-ABEND-TIMER VALUE 31. CMCOBOL
74 +000410 88 CM-SVC-ERROR-NO-TRUNC VALUE 32. CMCOBOL
75 +000420 88 CM-SVC-ERROR-PURGING VALUE 33. CMCOBOL
76 +000430 88 CM-SVC-ERROR-TRUNC VALUE 34. CMCOBOL
+000440* CMCOBOL

```

Figure F-6 (Part 3 of 8). COBOL/400 Inquiry Example – Local Program

77	+000450	01	DATA-RECEIVED	PIC 9(9) COMP-4.	CMCOBOL	
78	+000460	88	CM-NO-DATA-RECEIVED	VALUE 0.	CMCOBOL	
79	+000470	88	CM-DATA-RECEIVED	VALUE 1.	CMCOBOL	
80	+000480	88	CM-COMPLETE-DATA-RECEIVED	VALUE 2.	CMCOBOL	
81	+000490	88	CM-INCOMPLETE-DATA-RECEIVED	VALUE 3.	CMCOBOL	
	+000500*				CMCOBOL	
82	+000510	01	DEALLOCATE-TYPE	PIC 9(9) COMP-4.	CMCOBOL	
83	+000520	88	CM-DEALLOCATE-SYNC-LEVEL	VALUE 0.	CMCOBOL	
84	+000530	88	CM-DEALLOCATE-FLUSH	VALUE 1.	CMCOBOL	
85	+000540	88	CM-DEALLOCATE-CONFIRM	VALUE 2.	CMCOBOL	
86	+000550	88	CM-DEALLOCATE-ABEND	VALUE 3.	CMCOBOL	
	+000560*				CMCOBOL	
87	+000570	01	ERROR-DIRECTION	PIC 9(9) COMP-4.	CMCOBOL	
88	+000580	88	CM-RECEIVE-ERROR	VALUE 0.	CMCOBOL	
89	+000590	88	CM-SEND-ERROR	VALUE 1.	CMCOBOL	
	+000600*				CMCOBOL	
90	+000610	01	FILL	PIC 9(9) COMP-4.	CMCOBOL	
91	+000620	88	CM-FILL-LL	VALUE 0.	CMCOBOL	
92	+000630	88	CM-FILL-BUFFER	VALUE 1.	CMCOBOL	
	+000640*				CMCOBOL	
93	+000650	01	LOG-DATA	PIC X(512).	CMCOBOL	
	+000660*		0-512 BYTES		CMCOBOL	
	+000670*				CMCOBOL	
94	+000680	01	LOG-DATA-LENGTH	PIC 9(9) COMP-4.	CMCOBOL	
	+000690*				CMCOBOL	
95	+000700	01	MODE-NAME	PIC X(8).	CMCOBOL	
	+000710*		0-8 BYTES		CMCOBOL	
	+000720*				CMCOBOL	
96	+000730	01	MODE-NAME-LENGTH	PIC 9(9) COMP-4.	CMCOBOL	
	+000740*				CMCOBOL	
97	+000750	01	PARTNER-LU-NAME	PIC X(17).	CMCOBOL	
	+000760*		1-17 BYTES		CMCOBOL	
	+000770*				CMCOBOL	
98	+000780	01	PARTNER-LU-NAME-LENGTH	PIC 9(9) COMP-4.	CMCOBOL	
	+000790*				CMCOBOL	
99	+000800	01	PREPARE-TO-RECEIVE-TYPE	PIC 9(9) COMP-4.	CMCOBOL	
100	+000810	88	CM-PREP-TO-RECEIVE-SYNC-LEVEL	VALUE 0.	CMCOBOL	
101	+000820	88	CM-PREP-TO-RECEIVE-FLUSH	VALUE 1.	CMCOBOL	
102	+000830	88	CM-PREP-TO-RECEIVE-CONFIRM	VALUE 2.	CMCOBOL	
STMT	SEQNBR	-A 1	B.+. . . .2. . . .+. . . .3. . . .+. . . .4. . . .+. . . .5. . . .+. . . .6. . . .+. . . .7. . IDENTFCN	S	COPYNAME	CHG DATE
	+000840*				CMCOBOL	
103	+000850	01	RECEIVED-LENGTH	PIC 9(9) COMP-4.	CMCOBOL	
	+000860*				CMCOBOL	
104	+000870	01	RECEIVE-TYPE	PIC 9(9) COMP-4.	CMCOBOL	
105	+000880	88	CM-RECEIVE-AND-WAIT	VALUE 0.	CMCOBOL	
106	+000890	88	CM-RECEIVE-IMMEDIATE	VALUE 1.	CMCOBOL	
	+000900*				CMCOBOL	
107	+000910	01	REQUESTED-LENGTH	PIC 9(9) COMP-4.	CMCOBOL	
	+000920*				CMCOBOL	
108	+000930	01	REQUEST-TO-SEND-RECEIVED	PIC 9(9) COMP-4.	CMCOBOL	
109	+000940	88	CM-REQ-TO-SEND-NOT-RECEIVED	VALUE 0.	CMCOBOL	
110	+000950	88	CM-REQ-TO-SEND-RECEIVED	VALUE 1.	CMCOBOL	
	+000960*				CMCOBOL	
111	+000970	01	RETURN-CONTROL	PIC 9(9) COMP-4.	CMCOBOL	
112	+000980	88	CM-WHEN-SESSION-ALLOCATED	VALUE 0.	CMCOBOL	
113	+000990	88	CM-IMMEDIATE	VALUE 1.	CMCOBOL	
	+001000*				CMCOBOL	
114	+001010	01	SEND-LENGTH	PIC 9(9) COMP-4.	CMCOBOL	
	+001020*				CMCOBOL	
115	+001030	01	SEND-TYPE	PIC 9(9) COMP-4.	CMCOBOL	
116	+001040	88	CM-BUFFER-DATA	VALUE 0.	CMCOBOL	
117	+001050	88	CM-SEND-AND-FLUSH	VALUE 1.	CMCOBOL	
118	+001060	88	CM-SEND-AND-CONFIRM	VALUE 2.	CMCOBOL	
119	+001070	88	CM-SEND-AND-PREP-TO-RECEIVE	VALUE 3.	CMCOBOL	
120	+001080	88	CM-SEND-AND-DEALLOCATE	VALUE 4.	CMCOBOL	
	+001090*				CMCOBOL	

Figure F-6 (Part 4 of 8). COBOL/400 Inquiry Example — Local Program

```

121 +001100 01 STATUS-RECEIVED          PIC 9(9) COMP-4.          CMCOBOL
122 +001110      88 CM-NO-STATUS-RECEIVED VALUE 0.          CMCOBOL
123 +001120      88 CM-SEND-RECEIVED     VALUE 1.          CMCOBOL
124 +001130      88 CM-CONFIRM-RECEIVED   VALUE 2.          CMCOBOL
125 +001140      88 CM-CONFIRM-SEND-RECEIVED VALUE 3.      CMCOBOL
126 +001150      88 CM-CONFIRM-DEALLOC-RECEIVED VALUE 4.    CMCOBOL
    +001160*                                           CMCOBOL
127 +001170 01 SYNC-LEVEL                PIC 9(9) COMP-4.          CMCOBOL
128 +001180      88 CM-NONE                VALUE 0.          CMCOBOL
129 +001190      88 CM-CONFIRM              VALUE 1.          CMCOBOL
    +001200*                                           CMCOBOL
130 +001210 01 SYM-DEST-NAME              PIC X(8).              CMCOBOL
    +001220*                                           CMCOBOL
131 +001230 01 TP-NAME                    PIC X(64).             CMCOBOL
    +001240*      1-64 BYTES                      CMCOBOL
    +001250*                                           CMCOBOL
132 +001260 01 TP-NAME-LENGTH            PIC 9(9) COMP-4.      CMCOBOL
    001270
133 001280 PROCEDURE DIVISION.
    001290
    001300 START-PROGRAM SECTION.
    001310
    001320 START-PROGRAM-PARAGRAPH.
    001330
    001340*****
    001350* START OF PROGRAM                               *
    001360*                                               *
    001370* Files are opened, a conversation with the     *
    001380* remote program is started, and the part inquiry *
STMT SEQNBR -A 1 B...2...3...4...5...6...7..IDENTFCN S COPYNAME  CHG DATE
    001390* screen is displayed.  Inquiries are handled until *
    001400* the user presses the F3=Exit key, in which case *
    001410* the conversation will be ended and the program will end. *
    001420*****
    001430
3 134 001440 PERFORM OPEN-FILES.
135 001450 PERFORM START-CONVERSATION.
    001460
136 001470 PERFORM GET-CUST-NUM.
    001480
4 137 001490 PERFORM HANDLE-INQUIRY UNTIL CMD3-ON.
    001500
138 001510 SET CM-DEALLOCATE-FLUSH TO TRUE.
5 139 001520 PERFORM CLEAN-UP.
    001530
    001540
    001550*****
    001560* "OPEN-FILES" routine.                             *
    001570*                                               *
    001580* This routine opens the display file.             *
    001590*****
    001600
6 001610 OPEN-FILES.
140 001620 OPEN I-O T8189DSP.
141 001630 MOVE SPACES TO DSPREC.
7 142 001640 MOVE ZEROS TO DSPF-INDIC-AREA.
    001650
    001660*****
    001670* "START-CONVERSATION" routine.                     *
    001680*                                               *
    001690* This routine establishes a conversation with the *
    001700* remote program, and also sets various conversation *
    001710* characteristics.                                   *
    001720*****
    001730
8 001740 START-CONVERSATION.
    001750
143 001760 SET CM-DEALLOCATE-ABEND TO TRUE
    001770
    001780*****

```

Figure F-6 (Part 5 of 8). COBOL/400 Inquiry Example – Local Program

```

001790* The CMINIT call is issued to initialize various          *
001800* conversation characteristics.                               *
001810*****
 9 144 001820     MOVE "T8189CSI" TO SYM-DEST-NAME.
145 001830     CALL "CMINIT" USING CONVERSATION-ID
001840             SYM-DEST-NAME
001850             CM-RETCODE.
146 001860     IF CM-OK
147 001870         SET CONVERSATION-INITIALIZED TO TRUE
001880     ELSE
148 001890         PERFORM CLEAN-UP.
001900
001910*****
001920* The Set_TP_Name (CMSTPN) call is issued so that the        *
001930* TP_name conversation characteristic is set to the          *
STMT SEQNBR -A 1 B. ....2....3....4....5....6....7..IDENTFCN S COPYNAME  CHG DATE
001940* remote program.                                          *
001950*****
149 001960     MOVE "T8189CLT" TO TP-NAME.
150 001970     MOVE 8 TO TP-NAME-LENGTH.
10 151 001980     CALL "CMSTPN" USING CONVERSATION-ID
001990             TP-NAME
002000             TP-NAME-LENGTH
002010             CM-RETCODE.
002020
002030*****
002040* The Set_Sync_Level (CMSL) call is issued so that the      *
002050* sync_level conversation characteristic is set to          *
002060* CM_CONFIRH.                                              *
002070*****
152 002090     SET CM-CONFIRM TO TRUE.
11 153 002100     CALL "CMSL" USING CONVERSATION-ID
002110             SYNC-LEVEL
002120             CM-RETCODE.
002130
002140*****
002150* The CMALLC call is issued so that a conversation can be    *
002160* started using the conversation_ID previously assigned by    *
002170* the CMINIT call.                                          *
002180*****
12 154 002190     CALL "CMALLC" USING CONVERSATION-ID
002200             CM-RETCODE.
155 002210     IF NOT CM-OK
156 002220         PERFORM CLEAN-UP.
002230
002240*****
002250* The Set_Send_Type (CMSST) call is issued so that          *
002260* the send_type conversation characteristic is set to        *
002270* CM_SEND_AND_PREP_TO_RECEIVE.                              *
002280*****
157 002290     SET CM-SEND-AND-PREP-TO-RECEIVE TO TRUE.
13 158 002300     CALL "CMSST" USING CONVERSATION-ID
002310             SEND-TYPE
002320             CM-RETCODE.
002330
002340
002350*****
002360* "HANDLE-INQUIRY" routine.                                   *
002370*
002380* This is the main loop of the program. Process the part    *
002390* number keyed in by the user until F3 (CMD3) is pressed.   *
002400*****
14 002410
002420 HANDLE-INQUIRY.
002430
002440*****
002450* The part number that the user has requested information    *
002460* for is sent to the remote program using the CMSEND call.  *
002470*****
159 002480     MOVE 5 TO SEND-LENGTH.

```

Figure F-6 (Part 6 of 8). COBOL/400 Inquiry Example – Local Program

```

STMT SEQNBR -A 1 B.+. . . .2. . . .+. . . .3. . . .+. . . .4. . . .+. . . .5. . . .+. . . .6. . . .+. . . .7. . IDENTFCN S COPYNAME CHG DATE
15 160 002490 CALL "CMSEND" USING CONVERSATION-ID
    002500 PARTN OF PROMPT-I
    002510 SEND-LENGTH
    002520 REQUEST-TO-SEND-RECEIVED
    002530 CM-RETCODE.
    002540
    002550*****
    002560* The CMRCV call is issued to receive the response *
    002570* from the remote program (the response can either be *
    002580* an error message or the part description, depending *
    002590* on whether the part was found or not). *
    002600*****
16 161 002610 IF CM-OK
    162 002620 MOVE SPACES TO ERRORL
    163 002630 MOVE 25 TO REQUESTED-LENGTH
    164 002640 CALL "CMRCV" USING CONVERSATION-ID
        002650 PARTD OF PROMPT-O
        002660 REQUESTED-LENGTH
        002670 DATA-RECEIVED
        002680 RECEIVED-LENGTH
        002690 STATUS-RECEIVED
        002700 REQUEST-TO-SEND-RECEIVED
        002710 CM-RETCODE
    002720 ELSE
    165 002730 IF CM-PROGRAM-ERROR-PURGING
    166 002740 MOVE 40 TO REQUESTED-LENGTH
    167 002750 CALL "CMRCV" USING CONVERSATION-ID
        002760 ERRORL OF PROMPT-O
        002770 REQUESTED-LENGTH
        002780 DATA-RECEIVED
        002790 RECEIVED-LENGTH
        002800 STATUS-RECEIVED
        002810 REQUEST-TO-SEND-RECEIVED
        002820 CM-RETCODE
    168 002830 MOVE SPACES TO PARTD
    002840 ELSE
    169 002850 PERFORM CLEAN-UP.
    002860
    170 002870 IF CM-OK AND
        002880 CM-COMPLETE-DATA-RECEIVED AND
        002890 CM-SEND-RECEIVED
        002900 NEXT SENTENCE
    002910 ELSE
    171 002920 PERFORM CLEAN-UP.
    002930
17 172 002940 PERFORM GET-CUST-NUM.
    002950
    002960*****
    002970* "GET-CUST-NUM" routine. *
    002980* *
    002990* Get a customer number from the display. *
    003000*****
    003010
18 003020 GET-CUST-NUM.
    003030
STMT SEQNBR -A 1 B.+. . . .2. . . .+. . . .3. . . .+. . . .4. . . .+. . . .5. . . .+. . . .6. . . .+. . . .7. . IDENTFCN S COPYNAME CHG DATE
173 003040 WRITE DSPREC FORMAT IS "PROMPT".
174 003050 READ T8189DSP INDICATORS ARE DSPF-INDIC-AREA.
    003060
    003070*****
    003080* "CLEAN-UP" routine. *
    003090* *
    003100* The following code handles the end-of-program processing. *
    003110* This includes the ending of the conversation with *
    003120* the remote system (if conversation is active), and the *
    003130* closing of opened files. *
    003140*****
    003150

```

Figure F-6 (Part 7 of 8). COBOL/400 Inquiry Example — Local Program


```

19 003160 CLEAN-UP.
003170
20 175 003180     IF CONVERSATION-INITIALIZED AND NOT
003190         (CM-ALLOCATE-FAILURE-RETRY OR
003200          CM-ALLOCATE-FAILURE-NO-RETRY OR
003210          CM-DEALLOCATED-ABEND OR
003220          CM-DEALLOCATED-NORMAL OR
003230          CM-PRODUCT-SPECIFIC-ERROR OR
003240          CM-RESOURCE-FAILURE-RETRY OR
003250          CM-RESOURCE-FAILURE-NO-RETRY)
003260*****
003270* The deallocated_type has been previously set to either *
003280* CM_DEALLOCATE_FLUSH or CM_DEALLOCATE_ABEND. *
003290*****
176 003300         CALL "CMSDT" USING CONVERSATION-ID
003310             DEALLOCATE-TYPE
003320             CM-RETCODE
003330
177 003340         CALL "CMDEAL" USING CONVERSATION-ID
003350             CM-RETCODE
003360     END-IF.
003370
21 178 003380     CLOSE T8189DSP.
003390
179 003400     STOP RUN.
          * * * * *   E N D   O F   S O U R C E   * * * * *

STMT
*  MSGID: LBL0904 SEVERITY: 00 SEQNBR:
  Message . . . . : Unexpected source member type.
          * * * * *   E N D   O F   M E S S A G E S   * * * * *
          Message Summary
Total      Info(0-4)  Warning(5-19)  Error(20-29)  Severe(30-39)  Terminal(40-99)
   1         1         0             0             0             0
Source records read . . . . . : 340
Copy records read . . . . . : 137
Copy members processed . . . . . : 2
Sequence errors . . . . . : 0
Highest severity message issued . . : 0
  LBL0901 00 Program T8189CLS created in library APPCLIB.
          * * * * *   E N D   O F   C O M P I L A T I O N   * * * * *

```

Figure F-6 (Part 8 of 8). COBOL/400 Inquiry Example – Local Program

COBOL/400 Remote Program for Inquiry Application (Example 2)

The following explains the structure of the COBOL/400 remote program that handles requests sent by the partner program.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure F-7 on page F-29.

Note: On any type of error that is not expected (for example, an unexpected CPI Communications *return_code* on a call), the session is ended and the program ends.

1 The file division section defines the files used in the program.

T8189DB is the database file that contains the valid part numbers and part descriptions.

2 The COPY statement COPY CMCOBOL IN QLBL-QILBINC places the contents of the AS/400-supplied CPI Communications pseudonym file CMCOBOL in the program.

3 The OPEN-FILES, START-CONVERSATION, and GET-CUST-NUM routines are called to open files used by the program, start a conversation with the partner program, and wait on a request by the partner program, respectively.

4 The program loops until there are no more requests to process, or until an error occurs in the transaction with the partner program.

5 The CLEAN-UP routine is called to perform end-of-program processing.

6 The OPEN-FILES routine opens the database file.

7 The START-CONVERSATION routine establishes a conversation with the partner program and sets various conversation characteristics.

8 The Accept_Conversation (CMACCP) call is issued so that a conversation can be started with the partner program.

9 The Set_Send_Type (CMSST) call is issued so that the *send_type* conversation characteristic is set to CM_SEND_AND_PREP_TO_RECEIVE.

10 The Set_Prepare_To_Receive_Type (CMSPTR) call is issued so that the *prepare_to_receive_type* conversation characteristic is set to CM_PREP_TO_RECEIVE_FLUSH.

11 The HANDLE-INQUIRY routine contains the body of the loop that handles requests from the partner program.

12 A search of the database file is performed using the part number received from the partner program as the key.

13 If the part number is found in the database file, the Confirmed (CMCFMD) call is issued. As a result, a positive response to the received confirmation request is sent to the partner program.

14 If the part number is not found in the database file, the Send_Error (CMSERR) call is issued. As a result, a negative response to the received confirmation request is sent to the partner program.

15 The Send_Data (CMSSEND) call is issued. The data sent (set previously) is either an error message (if the part was not found) or the part description

(if the part is found). The CMSEND call is issued with the following conversation characteristics (the conversation characteristics were set in START-CONVERSATION): a *send_type* of CM_SEND_AND_PREP_TO_RECEIVE and a *prepare_to_receive_type* of CM_PREP_TO_RECEIVE_FLUSH. Setting the conversation characteristics to these values flushes the data and changes the data flow direction from send to receive.

16 The GET-CUST-NUM routine is called to wait on a request by the partner program.

17 The GET-CUST-NUM routine waits for a request from the partner program by issuing the Receive (CMRCV) call.

Note: A transaction is processed if all the data is received with a turnaround indication, and the partner program requested confirmation. This is indicated by the following:

- The value of the *data_received* variable on the CMRCV call is CM_COMPLETE_DATA_RECEIVED
- The value of the *status_received* variable on the CMRCV call is CM_CONFIRM_SEND_RECEIVED

18 The CLEAN-UP routine performs end-of-program processing.

19 If the conversation is still active, it is assumed that an error was detected. The *deallocate_type* is set to CM_DEALLOCATE_ABEND by issuing the Set_Deallocate_Type (CMSDT) call, followed by a call to Deallocate (CMDEAL) to end the conversation abnormally.

20 The database file is closed.

```

Program . . . . . : T8189CLT
Library . . . . . : APPCLIB
Source file . . . . . : QATTCBL
Library . . . . . : QUSRTOOL
Source member . . . . . : T8189CLT      09/26/90 08:27:02
Generation severity level . . . . . : 29
Text 'description' . . . . . : *BLANK
Source listing options . . . . . : *NONE
Generation options . . . . . : *NONE
Message limit:
  Number of messages . . . . . : *NOMAX
  Message limit severity . . . . . : 29
Print file . . . . . : QSYSPRT
Library . . . . . : *LIBL
FIPS flagging . . . . . : *NOFIPS *NOSEG *NODEB *NOOBSOLETE
SAA flagging . . . . . : *NOFLAG
Flagging severity . . . . . : 0
Replace program . . . . . : *YES
Target release . . . . . : *CURRENT
User profile . . . . . : *USER
Authority . . . . . : *LIBCRTAUT
Compiler . . . . . : IBM AS/400 COBOL/400
STMT SEQNBR -A 1 B.+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME  CHG DATE
  1 000010 IDENTIFICATION DIVISION.
    000020
  2 000030 PROGRAM-ID. T8191CLT.
    000040
    000050*****
    000060* Program name.....: T8189CLT *
    000070* Program description.: CPI Communications remote program *
    000080* Language.....: COBOL/400 *
    000090* *
    000100* This program accepts the incoming conversation by *
    000110* issuing an Accept_Conversation (CMACCP) call. It then *
    000120* issues a Receive (CMRCV) call to receive the part number *
    000130* from the remote system. When the CMRCV call completes, *
    000140* the status_received value will be CM_CONFIRM_SEND. The *
    000150* database file T8189DB is searched for the received part *
    000160* number. If the part number is found, the Confirmed *
    000170* (CMCFMD) call is issued, followed by a Send_Data *
    000180* (CMSEND) call (the data sent is the part description *
    000190* corresponding to the part number retrieved from the *
    000200* database file). However, if the part number is not *
    000210* found, the Send_Error (CMSERR) call is issued, followed *
    000220* by a CMSEND call (the data sent is a message describing *
    000230* the error). The CMSEND call sending either the part *
    000240* description or the error message is issued with a *
    000250* send_type conversation characteristic of *
    000260* CM_SEND_AND_PREP_TO_RECEIVE and a *
    000270* prepare_to_receive_type conversation characteristic of *
    000280* CM_PREP_TO_RECEIVE_FLUSH. Setting the conversation *
    000290* characteristics to these values results in the flushing *
    000300* of the data, and the changing of the data flow *
    000310* direction. The partner program can send more inquiries. *
    000320* *
    000330* This program will continue to handle inquiries from the *
    000340* partner program until a return_code that is not CM_OK *
    000350* is received. Then the program ends. *
    000360* *
    000370* NOTE 1: If an unexpected return_code value is received on *
    000380* any of the CPI Communications calls, the *
    000390* program will abnormally end the conversation *
    000400* with a deallocate_type of CM_DEALLOCATED_ABEND, *
    000410* and program processing will end. *
    000420* *
    000430* NOTE 2: On the CMRCV call, if the data_received *
    000440* parameter value does not indicate *
    000450* CM_COMPLETE_DATA_RECEIVED, or if the *
    000460* status_received parameter value does not indicate *
    000470* CM_CONFIRM_SEND_RECEIVED, the program will *

```

Figure F-7 (Part 1 of 7). COBOL/400 Inquiry Example – Remote Program

```

000480*          abnormally end the conversation with a          *
000490*          deallocate_type of CM_DEALLOCATED_ABEND,      *
000500*          and program processing will end.                *
000510*                                                         *
000520* NOTE 3: This program can be started by ANY of the     *
000530*          "local" program examples in the APPC Programmer's *
000540*          Guide.                                           *
000550*                                                         *
STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME CHG DATE
000560*****
000570
3 000580 ENVIRONMENT DIVISION.
000590
4 000600 CONFIGURATION SECTION.
000610
5 000620 SOURCE-COMPUTER. IBM-AS400.
6 000630 OBJECT-COMPUTER. IBM-AS400.
7 000640 SPECIAL-NAMES.
000650
8 000660 INPUT-OUTPUT SECTION.
000670
9 000680 FILE-CONTROL.
000690
10 000700      SELECT T8189DB ASSIGN TO DATABASE-T8189DB
11 000710          ORGANIZATION IS INDEXED
12 000720          ACCESS IS RANDOM
13 000730          RECORD KEY IS ITEMNH.
000740
14 000750 DATA DIVISION.
000760
15 000770 FILE SECTION.
000780
000790
000800*****
000810* File description for the database file.                *
000820*****
000830
1 16 000840 FD T8189DB
17 000850 LABEL RECORDS ARE STANDARD.
18 000860 01 DBREC.
19 000870 COPY DDS-ALL-FORMATS OF T8189DB.
20 +000001 05 T8189DB-RECORD PIC X(30).
+000002* I-O FORMAT:DBRCD FROM FILE T8189DB OF LIBRARY APPCLIB
+000003*
+000004* USER SUPPLIED KEY BY RECORD KEY CLAUSE
21 +000005 05 DBRCD REDEFINES T8189DB-RECORD.
22 +000006 06 ITEMNH PIC X(5).
23 +000007 06 ITEM D PIC X(25).
000880
24 000890 WORKING-STORAGE SECTION.
25 000900 01 SEND-BUFFER PIC X(40).
000910
26 000920 01 NOT-FND-MSG PIC X(40)
27 000930 VALUE "The requested part was not found. ".
000940
28 000950 01 ERROR-FND PIC X.
000960
2 29 000970 COPY CMCOBOL IN QLBL-QILBINC.
+000010*COPY CMCOBOL
+000020*****
+000030* NOTE: BUFFER MUST BE DEFINED IN WORKING STORAGE *
+000040*****
+000050*
30 +000060 01 CONVERSATION-ID PIC X(8).

```

```

<-ALL-FMTS
<-ALL-FMTS
<-ALL-FMTS
<-ALL-FMTS
<-ALL-FMTS
<-ALL-FMTS
<-ALL-FMTS

```

```

CMCOBOL
CMCOBOL
CMCOBOL
CMCOBOL
CMCOBOL
CMCOBOL

```

Figure F-7 (Part 2 of 7). COBOL/400 Inquiry Example — Remote Program

```

STMT SEQNBR -A 1 B...2...3...4...5...6...7..IDENTFCN S COPYNAME CHG DATE
+000070* CMCOBOL
31 +000080 01 CONVERSATION-TYPE PIC 9(9) COMP-4. CMCOBOL
32 +000090 88 CM-BASIC-CONVERSATION VALUE 0. CMCOBOL
33 +000100 88 CM-MAPPED-CONVERSATION VALUE 1. CMCOBOL
+000110* CMCOBOL
34 +000120 01 CM-RETCODE PIC 9(9) COMP-4. CMCOBOL
+000130* ==> RETURN-CODE IS A RESERVED WORD IN SOME <=== CMCOBOL
+000140* ==> VERSIONS OF COBOL <=== CMCOBOL
+000150* CMCOBOL
35 +000160 88 CM-OK VALUE 0. CMCOBOL
36 +000170 88 CM-ALLOCATE-FAILURE-NO-RETRY VALUE 1. CMCOBOL
37 +000180 88 CM-ALLOCATE-FAILURE-RETRY VALUE 2. CMCOBOL
38 +000190 88 CM-CONVERSATION-TYPE-MISMATCH VALUE 3. CMCOBOL
39 +000200 88 CM-PIP-NOT-SPECIFIED-CORRECTLY VALUE 5. CMCOBOL
40 +000210 88 CM-SECURITY-NOT-VALID VALUE 6. CMCOBOL
41 +000220 88 CM-SYNC-LVL-NOT-SUPPORTED-LU VALUE 7. CMCOBOL
42 +000230 88 CM-SYNC-LVL-NOT-SUPPORTED-PGM VALUE 8. CMCOBOL
43 +000240 88 CM-TPN-NOT-RECOGNIZED VALUE 9. CMCOBOL
44 +000250 88 CM-TP-NOT-AVAILABLE-NO-RETRY VALUE 10. CMCOBOL
45 +000260 88 CM-TP-NOT-AVAILABLE-RETRY VALUE 11. CMCOBOL
46 +000270 88 CM-DEALLOCATED-ABEND VALUE 17. CMCOBOL
47 +000280 88 CM-DEALLOCATED-NORMAL VALUE 18. CMCOBOL
48 +000290 88 CM-PARAMETER-ERROR VALUE 19. CMCOBOL
49 +000300 88 CM-PRODUCT-SPECIFIC-ERROR VALUE 20. CMCOBOL
50 +000310 88 CM-PROGRAM-ERROR-NO-TRUNC VALUE 21. CMCOBOL
51 +000320 88 CM-PROGRAM-ERROR-PURGING VALUE 22. CMCOBOL
52 +000330 88 CM-PROGRAM-ERROR-TRUNC VALUE 23. CMCOBOL
53 +000340 88 CM-PROGRAM-PARAMETER-CHECK VALUE 24. CMCOBOL
54 +000350 88 CM-PROGRAM-STATE-CHECK VALUE 25. CMCOBOL
55 +000360 88 CM-RESOURCE-FAILURE-NO-RETRY VALUE 26. CMCOBOL
56 +000370 88 CM-RESOURCE-FAILURE-RETRY VALUE 27. CMCOBOL
57 +000380 88 CM-UNSUCCESSFUL VALUE 28. CMCOBOL
58 +000390 88 CM-DEALLOCATED-ABEND-SVC VALUE 30. CMCOBOL
59 +000400 88 CM-DEALLOCATED-ABEND-TIMER VALUE 31. CMCOBOL
60 +000410 88 CM-SVC-ERROR-NO-TRUNC VALUE 32. CMCOBOL
61 +000420 88 CM-SVC-ERROR-PURGING VALUE 33. CMCOBOL
62 +000430 88 CM-SVC-ERROR-TRUNC VALUE 34. CMCOBOL
+000440* CMCOBOL
63 +000450 01 DATA-RECEIVED PIC 9(9) COMP-4. CMCOBOL
64 +000460 88 CM-NO-DATA-RECEIVED VALUE 0. CMCOBOL
65 +000470 88 CM-DATA-RECEIVED VALUE 1. CMCOBOL
66 +000480 88 CM-COMPLETE-DATA-RECEIVED VALUE 2. CMCOBOL
67 +000490 88 CM-INCOMPLETE-DATA-RECEIVED VALUE 3. CMCOBOL
+000500* CMCOBOL
68 +000510 01 DEALLOCATE-TYPE PIC 9(9) COMP-4. CMCOBOL
69 +000520 88 CM-DEALLOCATE-SYNC-LEVEL VALUE 0. CMCOBOL
70 +000530 88 CM-DEALLOCATE-FLUSH VALUE 1. CMCOBOL
71 +000540 88 CM-DEALLOCATE-CONFIRM VALUE 2. CMCOBOL
72 +000550 88 CM-DEALLOCATE-ABEND VALUE 3. CMCOBOL
+000560* CMCOBOL
73 +000570 01 ERROR-DIRECTION PIC 9(9) COMP-4. CMCOBOL
74 +000580 88 CM-RECEIVE-ERROR VALUE 0. CMCOBOL
75 +000590 88 CM-SEND-ERROR VALUE 1. CMCOBOL
+000600* CMCOBOL
76 +000610 01 FILL PIC 9(9) COMP-4. CMCOBOL
STMT SEQNBR -A 1 B...2...3...4...5...6...7..IDENTFCN S COPYNAME CHG DATE
77 +000620 88 CM-FILL-LL VALUE 0. CMCOBOL
78 +000630 88 CM-FILL-BUFFER VALUE 1. CMCOBOL
+000640* CMCOBOL
79 +000650 01 LOG-DATA PIC X(512). CMCOBOL
+000660* 0-512 BYTES CMCOBOL
+000670* CMCOBOL
80 +000680 01 LOG-DATA-LENGTH PIC 9(9) COMP-4. CMCOBOL
+000690* CMCOBOL
81 +000700 01 MODE-NAME PIC X(8). CMCOBOL
+000710* 0-8 BYTES CMCOBOL
+000720* CMCOBOL
82 +000730 01 MODE-NAME-LENGTH PIC 9(9) COMP-4. CMCOBOL

```

Figure F-7 (Part 3 of 7). COBOL/400 Inquiry Example – Remote Program

```

+000740*
83 +000750 01 PARTNER-LU-NAME          PIC X(17).
+000760* 1-17 BYTES
+000770*
84 +000780 01 PARTNER-LU-NAME-LENGTH   PIC 9(9) COMP-4.
+000790*
85 +000800 01 PREPARE-TO-RECEIVE-TYPE  PIC 9(9) COMP-4.
86 +000810 88 CM-PREP-TO-RECEIVE-SYNC-LEVEL VALUE 0.
87 +000820 88 CM-PREP-TO-RECEIVE-FLUSH  VALUE 1.
88 +000830 88 CM-PREP-TO-RECEIVE-CONFIRM VALUE 2.
+000840*
89 +000850 01 RECEIVED-LENGTH          PIC 9(9) COMP-4.
+000860*
90 +000870 01 RECEIVE-TYPE              PIC 9(9) COMP-4.
91 +000880 88 CM-RECEIVE-AND-WAIT       VALUE 0.
92 +000890 88 CM-RECEIVE-IMMEDIATE      VALUE 1.
+000900*
93 +000910 01 REQUESTED-LENGTH         PIC 9(9) COMP-4.
+000920*
94 +000930 01 REQUEST-TO-SEND-RECEIVED  PIC 9(9) COMP-4.
95 +000940 88 CM-REQ-TO-SEND-NOT-RECEIVED VALUE 0.
96 +000950 88 CM-REQ-TO-SEND-RECEIVED   VALUE 1.
+000960*
97 +000970 01 RETURN-CONTROL           PIC 9(9) COMP-4.
98 +000980 88 CM-WHEN-SESSION-ALLOCATED  VALUE 0.
99 +000990 88 CM-IMMEDIATE              VALUE 1.
+001000*
100 +001010 01 SEND-LENGTH              PIC 9(9) COMP-4.
+001020*
101 +001030 01 SEND-TYPE                PIC 9(9) COMP-4.
102 +001040 88 CM-BUFFER-DATA           VALUE 0.
103 +001050 88 CM-SEND-AND-FLUSH        VALUE 1.
104 +001060 88 CM-SEND-AND-CONFIRM      VALUE 2.
105 +001070 88 CM-SEND-AND-PREP-TO-RECEIVE VALUE 3.
106 +001080 88 CM-SEND-AND-DEALLOCATE   VALUE 4.
+001090*
107 +001100 01 STATUS-RECEIVED          PIC 9(9) COMP-4.
108 +001110 88 CM-NO-STATUS-RECEIVED    VALUE 0.
109 +001120 88 CM-SEND-RECEIVED        VALUE 1.
110 +001130 88 CM-CONFIRM-RECEIVED     VALUE 2.
111 +001140 88 CM-CONFIRM-SEND-RECEIVED VALUE 3.
112 +001150 88 CM-CONFIRM-DEALLOC-RECEIVED VALUE 4.
+001160*
STMT SEQNBR -A 1 B.....2.....3.....4.....5.....6.....7..IDENTFCN S COPYNAME CHG DATE
113 +001170 01 SYNC-LEVEL                PIC 9(9) COMP-4.
114 +001180 88 CM-NONE                   VALUE 0.
115 +001190 88 CM-CONFIRM                VALUE 1.
+001200*
116 +001210 01 SYM-DEST-NAME             PIC X(8).
+001220*
117 +001230 01 TP-NAME                   PIC X(64).
+001240* 1-64 BYTES
+001250*
118 +001260 01 TP-NAME-LENGTH            PIC 9(9) COMP-4.
000980
119 000990 PROCEDURE DIVISION.
001000
001010 START-PROGRAM SECTION.
001020
001030 START-PROGRAM-PARAGRAPH.
001040
001050*****
001060* START OF PROGRAM *
001070* *
001080* Files are opened, a conversation with the *
001090* remote program is started, and the part inquiry *
001100* processing starts. Inquiries are handled until a *
001110* CM_DEALLOCATED_NORMAL return_code is received. *
001120*****
001130

```

Figure F-7 (Part 4 of 7). COBOL/400 Inquiry Example – Remote Program

```

3 120 001140    PERFORM OPEN-FILES.
      121 001150    PERFORM START-CONVERSATION.
      122 001160    PERFORM GET-CUST-NUM.
           001170
4 123 001180    PERFORM HANDLE-INQUIRY UNTIL
           001190                NOT CM-OK.
           001200
5 124 001210    PERFORM CLEAN-UP.
           001220
           001230*****
           001240* "OPEN-FILES" routine.                *
           001250*                                         *
           001260* This routine opens the database file.    *
           001270*****
           001280
6 125 001290    OPEN-FILES.
           001300
           001310    OPEN I-O T8189DB.
           001320
           001330*****
           001340* "START-CONVERSATION" routine.          *
           001350*                                         *
           001360* This subroutine establishes a conversation with the *
           001370* remote program, and also sets various conversation *
           001380* characteristics.                       *
           001390*****
           001400
7 126 001410    START-CONVERSATION.
           001420
STMT SEQNBR -A 1 B..+...2....+...3....+...4....+...5....+...6....+...7..IDENTFCN S COPYNAME  CHG DATE
001430*****
001440* The CMACCP call is issued so that a conversation can be *
001450* started with the partner program.                    *
001460*****
8 126 001470    CALL "CMACCP" USING CONVERSATION-ID
           001480                CM-RETCODE.
      127 001490    IF CM-OK THEN
           001500        NEXT SENTENCE
           001510    ELSE
      128 001520    PERFORM CLEAN-UP.
           001530
           001540*****
           001550* The Set_Send_Type (CMSST) call is issued so that the *
           001560* send_type conversation characteristic is set to *
           001570* CM_SEND_AND_PREP_TO_RECEIVE.            *
           001580*****
      129 001590    SET CM-SEND-AND-PREP-TO-RECEIVE TO TRUE.
9 130 001600    CALL "CMSST" USING CONVERSATION-ID
           001610                SEND-TYPE
           001620                CM-RETCODE.
           001630
           001640*****
           001650* The Set_Prepare_To_Receive_Type (CMSPTR) call is issued *
           001660* so that the prepare_to_receive_type conversation *
           001670* characteristic is set to CM_PREP_TO_RECEIVE_FLUSH. *
           001680*****
      131 001690    SET CM-PREP-TO-RECEIVE-FLUSH TO TRUE.
10 132 001700    CALL "CMSPTR" USING CONVERSATION-ID
           001710                PREPARE-TO-RECEIVE-TYPE
           001720                CM-RETCODE.
           001730
           001740*****
           001750* "HANDLE-INQUIRY" routine.                *
           001760*                                         *
           001770* This is the main loop of the program. Process inquiry *
           001780* request until conversation is ended.      *
           001790*****
           001800
11 133 001810    HANDLE-INQUIRY.
           001820    MOVE "0" TO ERROR-FND.
           001830

```

Figure F-7 (Part 5 of 7). COBOL/400 Inquiry Example – Remote Program

```

001840*****
001850* A search of the database file is done using the part      *
001860* number as the key.                                         *
001870*****
12 134 001880      READ T8189DB FORMAT IS "DBRCD"
135 001890      INVALID KEY MOVE "1" TO ERROR-FND.
001900
001910*****
001920* If the part number is found, the CMCMD call is             *
001930* issued; otherwise, the CMSERR call is issued.              *
001940*****
136 001950      IF ERROR-FND = "0" THEN
137 001960      MOVE 25 TO SEND-LENGTH
138 001970      MOVE ITEMN TO SEND-BUFFER
STMT SEQNBR -A 1 B.+. . . . 2. . . . 3. . . . 4. . . . 5. . . . 6. . . . 7. IDENTFCN S COPYNAME CHG DATE
13 139 001980      CALL "CMCFMD" USING CONVERSATION-ID
001990      CM-RETCODE
002000      ELSE
140 002010      MOVE 40 TO SEND-LENGTH
141 002020      MOVE NOT-FND-MSG TO SEND-BUFFER
14 142 002030      CALL "CMSERR" USING CONVERSATION-ID
002040      REQUEST-TO-SEND-RECEIVED
002050      CM-RETCODE.
002060
143 002070      IF CM-OK THEN
002080      NEXT SENTENCE
002090      ELSE
144 002100      PERFORM CLEAN-UP.
002110
002120*****
002130* The CMSEND call is issued. The data sent (set previously) *
002140* is either an error message (if the part was not found)    *
002150* or the part description (if the part is found).           *
002160*****
002170
15 145 002180      CALL "CMSEND" USING CONVERSATION-ID
002190      SEND-BUFFER
002200      SEND-LENGTH
002210      REQUEST-TO-SEND-RECEIVED
002220      CM-RETCODE
146 002230      IF CM-OK THEN
147 002240      PERFORM GET-CUST-NUM
002250      ELSE
148 002260      PERFORM CLEAN-UP.
002270
002280
002290*****
002300* "GET-CUST-NUM" routine.                                     *
002310*                                                           *
002320* This subroutine waits for incoming data from the partner  *
002330* program by issuing the CMRCV call.                         *
002340*****
002350
16 002360 GET-CUST-NUM.
002370
149 002380      MOVE 5 TO REQUESTED-LENGTH.
002390
17 150 002400      CALL "CMRCV" USING CONVERSATION-ID
002410      ITEMN
002420      REQUESTED-LENGTH
002430      DATA-RECEIVED
002440      RECEIVED-LENGTH
002450      STATUS-RECEIVED
002460      REQUEST-TO-SEND-RECEIVED
002470      CM-RETCODE.
002480
151 002490      IF CM-OK THEN
152 002500      IF CM-COMPLETE-DATA-RECEIVED AND
002510      CM-CONFIRM-SEND-RECEIVED
002520      NEXT SENTENCE

```

Figure F-7 (Part 6 of 7). COBOL/400 Inquiry Example – Remote Program


```

STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME CHG DATE
002530 ELSE
153 002540 PERFORM CLEAN-UP
002550 ELSE
002560 NEXT SENTENCE.
002570
002580*****
002590* "CLEAN-UP" routine. *
002600* *
002610* The following code handles the end-of-program processing. *
002620* This includes the ending of the conversation with *
002630* the remote system (if conversation is active), and the *
002640* closing of opened files. *
002650*****
002660
18 002670 CLEAN-UP.
002680
19 154 002690 IF NOT
002700 (CM-ALLOCATE-FAILURE-RETRY OR
002710 CM-ALLOCATE-FAILURE-NO-RETRY OR
002720 CM-DEALLOCATED-ABEND OR
002730 CM-DEALLOCATED-NORMAL OR
002740 CM-PRODUCT-SPECIFIC-ERROR OR
002750 CM-RESOURCE-FAILURE-RETRY OR
002760 CM-RESOURCE-FAILURE-NO-RETRY)
002770
155 002780 SET CM-DEALLOCATE-ABEND TO TRUE
156 002790 CALL "CMSDT" USING CONVERSATION-ID
002800 DEALLOCATE-TYPE
002810 CM-RETCODE
002820
157 002830 CALL "CMDEAL" USING CONVERSATION-ID
002840 CM-RETCODE
002850 END-IF.
002860
20 158 002870 CLOSE T8189DB.
002880
159 002890 STOP RUN.
002900
* * * * * E N D O F S O U R C E * * * * *

STMT
* MSGID: LBL0904 SEVERITY: 00 SEQNBR:
Message . . . . : Unexpected source member type.
* * * * * E N D O F M E S S A G E S * * * * *
Message Summary
Total Info(0-4) Warning(5-19) Error(20-29) Severe(30-39) Terminal(40-99)
1 1 0 0 0 0
Source records read . . . . . : 290
Copy records read . . . . . : 133
Copy members processed . . . . . : 2
Sequence errors . . . . . : 0
Highest severity message issued . . : 0
LBL0901 00 Program T8189CLT created in library APPCLIB.
* * * * * E N D O F C O M P I L A T I O N * * * * *

```

Figure F-7 (Part 7 of 7). COBOL/400 Inquiry Example – Remote Program

RPG/400 Local Program for Inquiry Applications (Example 3)

The following explains the structure of the RPG/400 local program that sends requests to the partner program for processing.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure F-8 on page F-38.

Note: On any type of error that is not expected (for example, an unexpected CPI Communications *return_code* on a call), the session is ended and the program ends.

1 The files used in the program are described in the file specifications section.

T8189DSP is the name of the display device file that is used to request an entry from the work station and to display the results of the inquiry. T8189DSP uses the file-level keyword,

INDARA, which indicates that the file uses a separate indicator area.

All files are implicitly opened at the beginning of the RPG/400 program cycle.

The continuation lines for the file specification define the following:

KINFSR Specifies the subroutine FAIL is to be called when a file exception condition occurs.

2 The statement `/COPY QIRPG/QIRGINC,CMRPG` places the contents of the AS/400-supplied CPI Communications pseudonym file CMRPG in the program. Refer to "Using Pseudonyms When Writing Applications" on page 6-6 for further information on pseudonym files.

3 The STRCNV subroutine is called to start a conversation with the partner program. This is followed by the EXFMT operation, which allows the user to enter requests that are to be sent to the partner program.

4 The program loops until either F3 is pressed from the work station, which sets the indicator in the separate indicator area of the display file, or an error occurs in the transaction with the partner program.

5 The part number is sent to the partner program using the Send_Data (CMSEND) call. The CMSEND call is issued with the following conversation characteristics (the conversation characteristics were set in STRCNV): a *send_type* of `CM_SEND_AND_PREP_TO_RECEIVE`; a *prepare_to_receive_type* of `CM_PREP_TO_RECEIVE_SYNC_LEVEL`; and a *sync_level* of `CM_CONFIRM`. Setting the conversation characteristics to these values flushes the data, changes the data flow direction, and sends a confirmation request to the partner program. The partner program must now respond with a positive or negative response.

6 If the partner program responds with a positive response (*return_code* of `CM_OK`) to the confirmation request, a

Receive (CMRCV) call is issued to receive the part description. However, if the partner program responds with a negative response (*return_code* of `CM_PROGRAM_ERROR_PURGING`) to the confirmation request, a CMRCV call is issued to receive the error message.

7 The EXFMT operation is issued to display the information returned by the partner program and to obtain the next part number to be queried.

8 The following section of code performs the end-of-program processing. Indicator 85 determines if an error was detected.

9 If the conversation is active, the Deallocate (CMDEAL) call is issued to end the conversation.

Note: The *deallocate_type* (set previously in the program) is either set to `CM_DEALLOCATE_FLUSH` if no error was detected, or `CM_DEALLOCATE_ABEND` if an error was detected.

10 The last record indicator (LR) is set on. All files are implicitly closed, and the program ends.

11 The STRCNV subroutine establishes a conversation with the partner program.

12 The Initialize_Conversation (CMINIT) call is issued to initialize various conversation characteristics before the conversation is allocated.

Note: The *sym_dest_name* used is the side information object T8189CSI.

13 The Set_TP_Name (CMSTPN) call is issued so that the *TP_name* conversation characteristic is set to the remote program.

Note: The remote program that is to be started can be any of the remote programs in this appendix and in Appendix E, "Example Programs for Applications Using the ICF Application Program Interface" on page E-1.

14 The Set_Sync_Level (CMSSL) call is issued so that the *sync_level* conversation characteristic is set to `CM_CONFIRM`.

- 15** The Allocate (CMALLC) call is issued so that a conversation can be started using the *conversation_ID* previously assigned by the CMINIT call.
- 16** The Set_Send_Type (CMSST) call is issued so that the *send_type* conversation characteristic is set to CM_SEND_AND_PREP_TO_RECEIVE.
- 17** The FAIL subroutine gets control when a file exception or error occurs. The FAIL subroutine handles all file exceptions or errors by passing control to the section of code that performs the end-of-program processing (**8**).

```

Compiler . . . . . : IBM AS/400 RPG/400
Command Options:
  Program . . . . . : APPCLIB/T8189CRS
  Source file . . . . . : QUSRTOOL/QATTRPG
  Source member . . . . . : *PGM
Text not available for message RXT0073 file QRPMSG.
  Generation options . . . . . : *NOLIST *NOXREF *NOATR *NODUMP *NOOPTIMIZE
  Source listing indentation . . . . . : *NONE
  SAA flagging . . . . . : *NOFLAG
  Generation severity level . . . . . : 29
  Print file . . . . . : *LIBL/QSYSPRT
  Replace program . . . . . : *YES
  Target release . . . . . : *CURRENT
  User profile . . . . . : *USER
  Authority . . . . . : *LIBCRTAUT
  Text . . . . . : *SRCMBRTXT
  Phase trace . . . . . : *NO
  Intermediate text dump . . . . . : *NONE
  Snap dump . . . . . : *NONE
  Codelist . . . . . : *NONE
  Ignore decimal data error . . . . . : *NO
Actual Program Source:
  Member . . . . . : T8189CRS
  File . . . . . : QATTRPG
  Library . . . . . : QUSRTOOL
  Last Change . . . . . : 09/26/90 08:27:40
SEQUENCE NUMBER *...1....+...2....+...3....+...4....+...5....+...6....+...7...* IND DO LAST PAGE PROGRAM
                SOURCE LISTING USE NUM UPDATE LINE ID
10 H
11 FT8189DSPCF E WORKSTN
12 F KINFSR FAIL
13 I*****
14 I* Program name.....: T8189CRS *
15 I* Program description..: CPI Communications local program *
16 I* Language.....: RPG/400 *
17 I* *
18 I* This program invokes a program to handle part inquiry on *
19 I* the remote system. The Initialize_Conversation (CMINIT) *
20 I* call is issued using the sym_dest_name of 'T8189CSI'. *
21 I* The Allocate (CMALLC) call is issued, which results in *
22 I* the establishment of a conversation with the remote *
23 I* program. A display which prompts the user for the part *
24 I* number for which part information is requested is then *
25 I* displayed. When the user presses Enter, a Send_Data *
26 I* (CMSEND) call is issued (the data sent to the partner *
27 I* program is the part number). Note that the CMSEND call *
28 I* is issued with the following conversation *
29 I* characteristics: a send_type of *
30 I* CM_SEND_AND_PREP_TO_RECEIVE; a prepare_to_receive_type *
31 I* of CM_PREP_TO_RECEIVE_SYNC_LEVEL; and a sync_level of *
32 I* CM_CONFIRM. Setting the conversation characteristics to *
33 I* these values results in the flushing of the data, the *
34 I* changing of the data flow direction, and the sending of *
35 I* a confirmation request to the partner program. If the *
36 I* partner program responds with the Confirmed (CMCFMD) *
37 I* call to the confirmation request, the return_code *
38 I* parameter value on the CMSEND call will be set to CM_OK; *
39 I* the Receive (CMRCV) call is then issued to receive the *
40 I* part description. However, if the partner program *
41 I* responds with the Send_Error (CMSERR) call to the *
42 I* confirmation request, the return_code parameter value on *
43 I* the CMSEND call will be set to CM_PROGRAM_ERROR_PURGING; *
44 I* a CMRCV call is issued to receive the error message. *
45 I* *
46 I* The error message or part description (depending on *
47 I* whether the part number was found) will be displayed on *
48 I* the screen. *
49 I* *
50 I*

```

Figure F-8 (Part 1 of 10). RPG/400 Inquiry Example – Local Program

```

410 I* This program will continue to handle inquiries until the *
420 I* user presses the F3=Exit key. When F3=Exit is pressed, *
430 I* the Deallocate (CMDEAL) call is issued to end the *
440 I* conversation (note that the deallocate_type conversation *
450 I* characteristic is set to CM_DEALLOCATE_FLUSH), and *
460 I* program processing ends. *
470 I* *
SEQUENCE          IND   DO   LAST   PAGE   PROGRAM
NUMBER          *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE   NUM   UPDATE   LINE   ID
480 I* NOTE 1: If an unexpected return_code value is received on *
490 I*      any of the CPI Communications calls, the *
500 I*      program will abnormally end the conversation (if *
510 I*      it is still active), and program processing will *
520 I*      end. *
530 I* *
540 I* NOTE 2: On the CMRCV call, if the data_received *
550 I*      parameter value does not indicate *
560 I*      CM_COMPLETE_DATA_RECEIVED, or if the *
570 I*      status_received parameter value does not indicate *
580 I*      CM_SEND_RECEIVED, the program will abnormally end *
590 I*      the conversation (if it is still active), and *
600 I*      program processing will end. *
610 I* *
620 I* NOTE 3: This program can start ANY of the "remote" *
630 I*      program examples in the APPC Programmer's *
640 I*      Guide by changing the TPNM variable to the *
650 I*      remote program that is to be started. *
660 I* *
670 I*****
680 I*
690 I*****
700 I* Use the CPI Communications supplied pseudonyms. *
710 I*****
720 I*
2 730 I/COPY QIRPG/QIRGINC,CMRPG
A000000+ MEMBER CMRPG IN FILE QIRGINC LIBRARY QIRPG OPENED FOR /COPY.
A000010+ I*
A000020+ I* RPG INCLUDE FOR SAA COMMUNICATIONS SUPPORT
A000030+ I*
RECORD FORMAT(S): LIBRARY APPCLIB FILE T8189DSP.
EXTERNAL FORMAT PROMPT RPG NAME PROMPT
B000000+ INPUT FIELDS FOR RECORD PROMPT FILE T8189DSP FORMAT PROMPT.
B000001+ 1 5 PARTN
A000040+ ICMCONS DS
A000050+ I*****
A000060+ I* conversation_type values:
A000070+ I*
A000080+ I* CM_BASIC_CONVERSATION -- VALUE 0 (BASIC)
A000090+ I* CM_MAPPED_CONVERSATION -- VALUE 1 (MAPPED)
A000100+ I*
A000110+ I 0 C BASIC
A000120+ I 1 C MAPPED
A000130+ I*****
A000140+ I* data_received values:
A000150+ I*
A000160+ I* CM_NO_DATA_RECEIVED -- VALUE 0 (NODATA)
A000170+ I* CM_DATA_RECEIVED -- VALUE 1 (DATREC)
A000180+ I* CM_COMPLETE_DATA_RECEIVED -- VALUE 2 (COMDAT)
A000190+ I* CM_INCOMPLETE_DATA_RECEIVED -- VALUE 3 (INCDAT)
A000200+ I*
A000210+ I 0 C NODATA
A000220+ I 1 C DATREC
A000230+ I 2 C COMDAT

```

Figure F-8 (Part 2 of 10). RPG/400 Inquiry Example – Local Program

```

SEQUENCE          IND  DO  LAST  PAGE  PROGRAM
NUMBER *...1....+...2....+...3....+...4....+...5....+...6....+...7...*  USE  NUM  UPDATE  LINE  ID
A000240+ I          3          C      INCDAT
A000250+ I*****
A000260+ I* deallocate_type values:
A000270+ I*
A000280+ I*  CM_DEALLOCATE_SYNC_LEVEL      -- VALUE 0  (DESYNC)
A000290+ I*  CM_DEALLOCATE_FLUSH          -- VALUE 1  (DEFLUS)
A000300+ I*  CM_DEALLOCATE_CONFIRM        -- VALUE 2  (DECONF)
A000310+ I*  CM_DEALLOCATE_ABEND         -- VALUE 3  (DEABTY)
A000320+ I*
A000330+ I          0          C      DESYNC
A000340+ I          1          C      DEFLUS
A000350+ I          2          C      DECONF
A000360+ I          3          C      DEABTY
A000370+ I*****
A000380+ I* error_direction values:
A000390+ I*
A000400+ I*  CM_RECEIVE_ERROR              -- VALUE 0  (RCVERR)
A000410+ I*  CM_SEND_ERROR                -- VALUE 1  (SNDERR)
A000420+ I*
A000430+ I          0          C      RCVERR
A000440+ I          1          C      SNDERR
A000450+ I*****
A000460+ I* fill values:
A000470+ I*
A000480+ I*  CM_FILL_LL                    -- VALUE 0  (FILLL)
A000490+ I*  CM_FILL_BUFFER              -- VALUE 1  (FILBUF)
A000500+ I*
A000510+ I          0          C      FILLL
A000520+ I          1          C      FILBUF
A000530+ I*****
A000540+ I* prepare_to_receive_type values:
A000550+ I*
A000560+ I*  CM_PREP_TO_RECEIVE_SYNC_LEVEL -- VALUE 0  (PTRSL)
A000570+ I*  CM_PREP_TO_RECEIVE_FLUSH     -- VALUE 1  (PTRFLS)
A000580+ I*  CM_PREP_TO_RECEIVE_CONFIRM  -- VALUE 2  (PTRCON)
A000590+ I*
A000600+ I          0          C      PTRSL
A000610+ I          1          C      PTRFLS
A000620+ I          2          C      PTRCON
A000630+ I*****
A000640+ I* receive_type values:
A000650+ I*
A000660+ I*  CM_RECEIVE_AND_WAIT           -- VALUE 0  (RCVWAT)
A000670+ I*  CM_RECEIVE_IMMEDIATE        -- VALUE 1  (RCVIMM)
A000680+ I*
A000690+ I          0          C      RCVWAT
A000700+ I          1          C      RCVIMM
A000710+ I*****
A000720+ I* request_to_send_received values:
A000730+ I*
A000740+ I*  CM_REQ_TO_SEND_NOT_RECEIVED   -- VALUE 0  (RTSNOT)
A000750+ I*  CM_REQ_TO_SEND_RECEIVED     -- VALUE 1  (RTSREC)
A000760+ I*
A000770+ I          0          C      RTSNOT
SEQUENCE          IND  DO  LAST  PAGE  PROGRAM
NUMBER *...1....+...2....+...3....+...4....+...5....+...6....+...7...*  USE  NUM  UPDATE  LINE  ID
A000780+ I          1          C      RTSREC
A000790+ I*****
A000800+ I* return_code values:
A000810+ I*
A000820+ I*  CM_OK                          -- VALUE 0  (CMOK)
A000830+ I*  CM_ALLOCATE_FAILURE_NO_RETRY  -- VALUE 1  (ALFLNR)
A000840+ I*  CM_ALLOCATE_FAILURE_RETRY    -- VALUE 2  (ALFLRE)
A000850+ I*  CM_CONVERSATION_TYPE_MISMATCH -- VALUE 3  (CNVMIS)
A000860+ I*  CM_PIP_NOT_SPECIFIED_CORRECTLY -- VALUE 5  (PIPNSC)
A000870+ I*  CM_SECURITY_NOT_VALID        -- VALUE 6  (SECNVL)
A000880+ I*  CM_SYNC_LVL_NOT_SUPPORTED_LU  -- VALUE 7  (SLNSLU)
A000890+ I*  CM_SYNC_LVL_NOT_SUPPORTED_PGM -- VALUE 8  (SLNSP)

```

Figure F-8 (Part 3 of 10). RPG/400 Inquiry Example – Local Program

```

A000900+ I* CM_TPN_NOT_RECOGNIZED -- VALUE 9 (TPNAME)
A000910+ I* CM_TP_NOT_AVAILABLE_NO_RETRY -- VALUE 10 (TPNORE)
A000920+ I* CM_TP_NOT_AVAILABLE_RETRY -- VALUE 11 (TPRET)
A000930+ I* CM_DEALLOCATED_ABEND -- VALUE 17 (DEABND)
A000940+ I* CM_DEALLOCATED_NORMAL -- VALUE 18 (DENORM)
A000950+ I* CM_PARAMETER_ERROR -- VALUE 19 (PARERR)
A000960+ I* CM_PRODUCT_SPECIFIC_ERROR -- VALUE 20 (PRODER)
A000970+ I* CM_PROGRAM_ERROR_NO_TRUNC -- VALUE 21 (PENOTR)
A000980+ I* CM_PROGRAM_ERROR_PURGING -- VALUE 22 (PEPURG)
A000990+ I* CM_PROGRAM_ERROR_TRUNC -- VALUE 23 (PETRNC)
A001000+ I* CM_PROGRAM_PARAMETER_CHECK -- VALUE 24 (PEPCHK)
A001010+ I* CM_PROGRAM_STATE_CHECK -- VALUE 25 (STACHK)
A001020+ I* CM_RESOURCE_FAILURE_NO_RETRY -- VALUE 26 (RFNORE)
A001030+ I* CM_RESOURCE_FAILURE_RETRY -- VALUE 27 (RFRET)
A001040+ I* CM_UNSUCCESSFUL -- VALUE 28 (UNSUCC)
A001050+ I* CM_DEALLOCATED_ABEND_SVC -- VALUE 30 (DABSVC)
A001060+ I* CM_DEALLOCATED_ABEND_TIMER -- VALUE 31 (DABTIM)
A001070+ I* CM_SVC_ERROR_NO_TRUNC -- VALUE 32 (SVCENT)
A001080+ I* CM_SVC_ERROR_PURGING -- VALUE 33 (SVCEP)
A001090+ I* CM_SVC_ERROR_TRUNC -- VALUE 34 (SVCET)
A001100+ I*
A001110+ I 0 C CMOK
A001120+ I 1 C ALFLNR
A001130+ I 2 C ALFLRE
A001140+ I 3 C CNVHIS
A001150+ I 5 C PIPNSC
A001160+ I 6 C SECNVL
A001170+ I 7 C SLNSLU
A001180+ I 8 C SLNSP
A001190+ I 9 C TPNAME
A001200+ I 10 C TPNORE
A001210+ I 11 C TPRET
A001220+ I 17 C DEABND
A001230+ I 18 C DENORM
A001240+ I 19 C PARERR
A001250+ I 20 C PRODER
A001260+ I 21 C PENOTR
A001270+ I 22 C PEPURG
A001280+ I 23 C PETRNC
A001290+ I 24 C PEPCHK
A001300+ I 25 C STACHK
A001310+ I 26 C RFNORE
SEQUENCE
NUMBER *...1...+...2...+...3...+...4...+...5...+...6...+...7...* IND DO LAST PAGE PROGRAM
USE NUM UPDATE LINE ID
A001320+ I 27 C RFRET
A001330+ I 28 C UNSUCC
A001340+ I 30 C DABSVC
A001350+ I 31 C DABTIM
A001360+ I 32 C SVCENT
A001370+ I 33 C SVCEP
A001380+ I 34 C SVCET
A001390+ I*****
A001400+ I* return_control values:
A001410+ I*
A001420+ I* CM_WHEN_SESSION_ALLOCATED -- VALUE 0 (SESALL)
A001430+ I* CM_IMMEDIATE -- VALUE 1 (IMMED)
A001440+ I*
A001450+ I 0 C SESALL
A001460+ I 1 C IMMED
A001470+ I*****
A001480+ I* send_type values:
A001490+ I*
A001500+ I* CM_BUFFER_DATA -- VALUE 0 (BUFDAT)
A001510+ I* CM_SEND_AND_FLUSH -- VALUE 1 (SNDFLS)
A001520+ I* CM_SEND_AND_CONFIRM -- VALUE 2 (SND CNF)
A001530+ I* CM_SEND_AND_PREP_TO_RECEIVE -- VALUE 3 (SNDPTR)
A001540+ I* CM_SEND_AND_DEALLOCATE -- VALUE 4 (SNDDDEL)

```

Figure F-8 (Part 4 of 10). RPG/400 Inquiry Example – Local Program

```

A001550+ I*
A001560+ I      0      C      BUFDAT
A001570+ I      1      C      SNDFLS
A001580+ I      2      C      SNDCNF
A001590+ I      3      C      SNDPTR
A001600+ I      4      C      SNDDDEL
A001610+ I*****
A001620+ I* status_received values:
A001630+ I*
A001640+ I* CM_NO_STATUS_RECEIVED      -- VALUE 0 (NOSTAT)
A001650+ I* CM_SEND_RECEIVED          -- VALUE 1 (SNDREC)
A001660+ I* CM_CONFIRM_RECEIVED       -- VALUE 2 (CONRCV)
A001670+ I* CM_CONFIRM_SEND_RECEIVED  -- VALUE 3 (CONSND)
A001680+ I* CM_CONFIRM_DEALLOC_RECEIVED -- VALUE 4 (CONDEL)
A001690+ I*
A001700+ I      0      C      NOSTAT
A001710+ I      1      C      SNDREC
A001720+ I      2      C      CONRCV
A001730+ I      3      C      CONSND
A001740+ I      4      C      CONDEL
A001750+ I*****
A001760+ I* sync_level values:
A001770+ I*
A001780+ I* CM_NONE                   -- VALUE 0 (NONE)
A001790+ I* CM_CONFIRM                -- VALUE 1 (CONFRM)
A001800+ I*
A001810+ I      0      C      NONE
A001820+ I      1      C      CONFRM
740 I*
* 4110      4110-**
SEQUENCE
NUMBER *...1...+...2...+...3...+...4...+...5...+...6...+...7...*
750 ICMPARM DS
760 I      1 8 CONVID
770 I      B 9 120RTNCOD
780 I      B 13 160DATRCV
790 I      B 17 200DLCTYP
800 I      B 21 240RCVLEN
810 I      B 25 280REQLEN
820 I      B 29 320REQTSR
830 I      B 33 360SNDLEN
840 I      B 37 400SNDTYP
850 I      B 41 440STRRCV
860 I      B 45 480SYNLVL
870 I      49 56 SYMDST
880 I      57 64 TPNAM
890 I      B 65 680TPNLEN
900 C*****
910 C* START OF PROGRAM *
920 C* *
930 C* Files are implicitly opened, a conversation with the *
940 C* remote program is started, and the part inquiry *
950 C* screen is displayed. Inquiries are handled until *
960 C* the user presses the F3=Exit key, in which case *
970 C* the conversation will be ended and the program will end. *
980 C*****
990 C*
3 1000 C      EXSR STRCNV
1010 C      EXFMTPROMPT
1020 C*
4 1030 C      *IN99 DOWEQ'0'      B001
1040 C*
1050 C*****
1060 C* The part number that the user has requested information *
1070 C* for is sent to the remote program using the CMSEND call. *
1080 C*****
1090 C*
1100 C      Z-ADD5      SNDLEN      001
5 1110 C      CALL 'CMSEND'      001
1120 C      PARM      CONVID      001
1130 C      PARM      PARTN      001

```

Figure F-8 (Part 5 of 10). RPG/400 Inquiry Example – Local Program


```

1140 C          PARM          SNDLEN          001
1150 C          PARM          REQTSR          001
1160 C          PARM          RTNCOD          001
1170 C*
1180 C*****
1190 C* The CMRCV call is issued to receive the response      *
1200 C* from the remote program (the response can either be   *
1210 C* an error message or the part description, depending  *
1220 C* on whether the part was found or not).                *
1230 C*****
1240 C*
6 1250 C          RTNCOD    IFEQ CMOK          B002
1260 C          Z-ADD25      REQLEN          002
SEQUENCE          *...1...+...2...+...3...+...4...+...5...+...6...+...7...*   IND   DO   LAST   PAGE   PROGRAM
NUMBER          *...1...+...2...+...3...+...4...+...5...+...6...+...7...*   USE   NUM  UPDATE  LINE   ID
1270 C          CALL 'CMRCV'
1280 C          PARM          CONVID          002
1290 C          PARM          PARTD          002
1300 C          PARM          REQLEN          002
1310 C          PARM          DATRCV          002
1320 C          PARM          RCVLEN          002
1330 C          PARM          STSRCV          002
1340 C          PARM          REQTSR          002
1350 C          PARM          RTNCOD          002
1360 C          MOVE *BLANKS  ERRORL          002
1370 C          ELSE          X002
1380 C          RTNCOD    IFEQ PEPURG          B003
1390 C          Z-ADD40      REQLEN          003
1400 C          CALL 'CMRCV'
1410 C          PARM          CONVID          003
1420 C          PARM          ERRORL          003
1430 C          PARM          REQLEN          003
1440 C          PARM          DATRCV          003
1450 C          PARM          RCVLEN          003
1460 C          PARM          STSRCV          003
1470 C          PARM          REQTSR          003
1480 C          PARM          RTNCOD          003
1490 C          MOVE *BLANKS  PARTD          003
1500 C          ELSE          X003
1510 C          GOTO ENDCNV          003
1520 C          END          E003
1530 C          END          E002
1540 C*
1550 C          RTNCOD    IFNE CMOK          B002
1560 C          DATRCV    ORNE COMDAT          002
1570 C          STSRCV    ORNE SNDREC          002
1580 C          GOTO ENDCNV          002
1590 C          END          E002
7 1600 C          EXFMTPROMPT          001
1610 C*
1620 C          END          E001
1630 C*
1640 C*****
1650 C* The following code handles the end-of-program processing. *
1660 C* This includes the ending of the conversation with      *
1670 C* the remote system (if conversation is active), and    *
1680 C* the setting of the last record indicator.              *
1690 C*****
1700 C*
1710 C          Z-ADDDEFLUS  DLCTYP
1720 C*
8 1730 C          ENDCNV    TAG
1740 C*
9 1750 C          RTNCOD    IFNE ALFLNR          B001
1760 C          RTNCOD    ANDNEALFLRE          001
1770 C          RTNCOD    ANDNEDEABND          001
1780 C          RTNCOD    ANDNEDENORM          001
1790 C          RTNCOD    ANDNEPRODER          001
1800 C          RTNCOD    ANDNERFNORE          001

```

Figure F-8 (Part 6 of 10). RPG/400 Inquiry Example – Local Program

SEQUENCE NUMBER	*...1...+...2...+...3...+...4...+...5...+...6...+...7...*	IND USE	DO NUM	LAST UPDATE	PAGE LINE	PROGRAM ID
1810	C RTNCO					
1820	C*****					
1830	C* The deallocated_type has been previously set to either *					
1840	C* CM_DEALLOCATED_NORMAL or CM_DEALLOCATED_ABEND. *					
1850	C*****					
1860	C CALL 'CMSDT'		001			
1870	C PARM CONVID		001			
1880	C PARM DLCTYP		001			
1890	C PARM RTNCO		001			
1900	C CALL 'CMDEAL'		001			
1910	C PARM CONVID		001			
1920	C PARM RTNCO		001			
1930	C END		E001			
1940	C*					
1950	C ENDPGM TAG					
10 1960	C SETON LR	1				
1970	C*					
1980	C*****					
1990	C* "STRCNV" subroutine. *					
2000	C*					
2010	C* This subroutine establishes a conversation with the *					
2020	C* remote program, and also sets various conversation *					
2030	C* characteristics. *					
2040	C*****					
2050	C*					
11 2060	CSR STRCNV BEGSR					
2070	C*					
2080	C*****					
2090	C* Set the local deallocate_type parameter variable to *					
2100	C* CM_DEALLOCATE_ABEND. *					
2110	C*****					
2120	C*					
2130	C Z-ADDDEABTY DLCTYP					
2140	C*					
2150	C*****					
2160	C* The CMINIT call is issued to initialize various *					
2170	C* conversation characteristics. *					
2180	C*****					
2190	C*					
2200	CSR MOVE 'T8189CSI' SYMDST 8					
12 2210	CSR CALL 'CMINIT'					
2220	CSR PARM CONVID					
2230	CSR PARM SYMDST					
2240	CSR PARM RTNCO					
2250	CSR RTNCO CABNECMOK ENDPGM					
2260	C*					
2270	C*****					
2280	C* The Set_TP_Name (CMSTPN) call is issued so that the *					
2290	C* TP_name conversation characteristic is set to the *					
2300	C* remote program. *					
2310	C*****					
2320	C*					
2330	CSR MOVE 'T8189CRT' TPNAM 8					
2340	CSR Z-ADD8 TPNLEN					
SEQUENCE NUMBER	*...1...+...2...+...3...+...4...+...5...+...6...+...7...*	IND USE	DO NUM	LAST UPDATE	PAGE LINE	PROGRAM ID
13 2350	CSR CALL 'CMSTPN'					
2360	CSR PARM CONVID					
2370	CSR PARM TPNAM					
2380	CSR PARM TPNLEN					
2390	CSR PARM RTNCO					
2400	C*					

Figure F-8 (Part 7 of 10). RPG/400 Inquiry Example — Local Program

```

2410 C*****
2420 C* The Set_Sync_Level (CMSSL) call is issued so that the *
2430 C* sync_level conversation characteristic is set to *
2440 C* CM_CONFIRM. *
2450 C*****
2460 C*
14 2470 CSR          Z-ADDCONFRM  SYNLVL
2480 CSR          CALL 'CMSSL'
2490 CSR          PARM          CONVID
2500 CSR          PARM          SYNLVL
2510 CSR          PARM          RTNCOD
2520 C*
2530 C*****
2540 C* The CMALLC call is issued so that a conversation can be *
2550 C* started using the conversation_ID previously assigned by *
2560 C* the CMINIT call. *
2570 C*****
2580 C*
15 2590 CSR          CALL 'CMALLC'
2600 CSR          PARM          CONVID
2610 CSR          PARM          RTNCOD
2620 CSR          RTNCOD  CABNECHOK  ENDCNV
2630 C*
2640 C*****
2650 C* The Set_Send_Type (CMSST) call is issued so that *
2660 C* the send_type conversation characteristic is set to *
2670 C* CM_SEND_AND_PREP_TO_RECEIVE. *
2680 C*****
2690 C*
2700 CSR          Z-ADDSNDPTR  SNDTYP
16 2710 CSR          CALL 'CMSST'
2720 CSR          PARM          CONVID
2730 CSR          PARM          SNDTYP
2740 CSR          PARM          RTNCOD
2750 C*
2760 CSR          ENDSR
2770 C*
2780 C*****
2790 C* "FAIL" subroutine. *
2800 C* *
2810 C* This subroutine handles file exception/errors. *
2820 C*****
2830 C*
17 2840 CSR          FAIL      BEGSR
2850 CSR          GOTO ENDCNV
2860 CSR          ENDSR
C000000  OUTPUT FIELDS FOR RECORD PROMPT FILE T8189DSP FORMAT PROMPT.
C000001  PARTN      5 CHAR      5
SEQUENCE
NUMBER  *...1...+...2...+...3...+...4...+...5...+...6...+...7...*
C000002  PARTD      30 CHAR     25
C000003  ERRORL     70 CHAR     40
*****  END OF SOURCE *****
Additional Diagnostic Messages
* 7111  SOURCE FILE MEMBER HAS AN UNEXPECTED SOURCE TYPE.
* 7089  20  RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE T8189DSP.
Cross Reference
File and Record References:
FILE/RCD  DEV/RCD  REFERENCES (D=DEFINED)
01 T8189DSP  WORKSTN  20D
PROMPT      20D B000000  1010  1600  C000000
Field References:
FIELD     ATTR  REFERENCES (M=MODIFIED D=DEFINED)
*IN99    A(1)  1030
ALFLNR   CONST A001120D  1750
ALFLRE   CONST A001130D  1760
* 7031   BASIC  CONST A000110D
* 7031   BUFDAT CONST A001560D
* 7031   CHCONS DS(1)  A000040D

```

Figure F-8 (Part 8 of 10). RPG/400 Inquiry Example – Local Program

	CMOK	CONST	A001110D	1250	1550	2250	2620
* 7031	CMPARM	DS(68)	750D				
* 7031	CNMVIS	CONST	A001140D				
	COMDAT	CONST	A000230D	1560			
* 7031	CONDEL	CONST	A001740D				
	CONFRM	CONST	A001820D	2470			
* 7031	CONRCV	CONST	A001720D				
* 7031	CONSND	CONST	A001730D				
	CONVID	A(8)	760D	1120	1280	1410	1870
			1910	2220	2360	2490	2600
			2720				
* 7031	DABSVC	CONST	A001340D				
* 7031	DABTIM	CONST	A001350D				
	DATRCV	B(9,0)	780D	1310	1440	1560	
* 7031	DATREC	CONST	A000220D				
	DEABND	CONST	A001220D	1770			
	DEABTY	CONST	A000360D	2130			
* 7031	DECONF	CONST	A000350D				
	DEFLUS	CONST	A000340D	1710			
	DENORM	CONST	A001230D	1780			
* 7031	DESYNC	CONST	A000330D				
	DLCTYP	B(9,0)	790D	1710M	1880	2130M	
	ENDCNV	TAG	1510	1580	1730D	2620	2850
	ENDPGM	TAG	1950D	2250			
	ERRORL	A(40)	1360M	1420	C000003D		
	FAIL	BEGSR	20	2840D			
* 7031	FILBUF	CONST	A000520D				
* 7031	FILLL	CONST	A000510D				
* 7031	IMMED	CONST	A001460D				
* 7031	INCDAT	CONST	A000240D				
* 7031	MAPPED	CONST	A000120D				
* 7031	NODATA	CONST	A000210D				
* 7031	NONE	CONST	A001810D				
* 7031	NOSTAT	CONST	A001700D				
* 7031	PARERR	CONST	A001240D				
	PARTD	A(25)	1290	1490M	C000002D		
	PARTN	A(5)	B000001D	1130	C000001D		
* 7031	PENOTR	CONST	A001260D				
* 7031	PEPCHK	CONST	A001290D				
	PEPURG	CONST	A001270D	1380			
* 7031	PETRNC	CONST	A001280D				
* 7031	PIPNSC	CONST	A001150D				
	PRODER	CONST	A001250D	1790			
* 7031	PTRCON	CONST	A000620D				
* 7031	PTRFLS	CONST	A000610D				
* 7031	PTRSL	CONST	A000600D				
* 7031	RCVERR	CONST	A000430D				
* 7031	RCVIMM	CONST	A000700D				
	RCVLEN	B(9,0)	800D	1320	1450		
* 7031	RCVWAT	CONST	A000690D				
	REQLEN	B(9,0)	810D	1260M	1300	1390M	1430
	REQTSR	B(9,0)	820D	1150	1340	1470	
	RFNORE	CONST	A001310D	1800			
	RFRET	CONST	A001320D	1810			
	RTNCOD	B(9,0)	770D	1160	1250	1350	1380
			1480	1550	1750	1760	1770
			1780	1790	1800	1810	1890
			1920	2240	2250	2390	2510
			2610	2620	2740		
* 7031	RTSNOT	CONST	A000770D				
* 7031	RTSREC	CONST	A000780D				
* 7031	SECNVL	CONST	A001160D				
* 7031	SESALL	CONST	A001450D				
* 7031	SLNSLU	CONST	A001170D				
* 7031	SLNSP	CONST	A001180D				
* 7031	SNDCNF	CONST	A001580D				
* 7031	SNDDEL	CONST	A001600D				
* 7031	SNDERR	CONST	A000440D				
* 7031	SNDFLS	CONST	A001570D				
	SNDLEN	B(9,0)	830D	1100M	1140		
	SNDPTR	CONST	A001590D	2700			

Figure F-8 (Part 9 of 10). RPG/400 Inquiry Example – Local Program

```

      SNDREC      CONST  A001710D   1570
      SNDTYP      B(9,0)    840D   2700M   2730
* 7031 STACHK      CONST  A001300D
      STRCNV      BEGSR    1000    2060D
      STSRCV      B(9,0)    850D   1330    1460    1570
* 7031 SVCENT      CONST  A001360D
* 7031 SVCEP      CONST  A001370D
* 7031 SVCET      CONST  A001380D
      SYMDST      A(8)      870D   2200D   2230
      SYNLVL      B(9,0)    860D   2470M   2500
      TPNAM       A(8)      880D   2330D   2370
* 7031 TPNAME      CONST  A001190D
      TPNLEN      B(9,0)    890D   2340M   2380
* 7031 TPNORE      CONST  A001200D
* 7031 TPRET      CONST  A001210D
* 7031 UNSUCC      CONST  A001330D
      *BLANKS     LITERAL  1360    1490
      'CMALLC'    LITERAL  2590
      'CMDEAL'    LITERAL  1900
      'CMINIT'    LITERAL  2210
      'CMRCV'     LITERAL  1270    1400
      'CMSDT'     LITERAL  1860
      'CMSEND'    LITERAL  1110
      'CMSSL'     LITERAL  2480
      'CMSST'     LITERAL  2710
      'CMSTPN'    LITERAL  2350
      'T8189CRT'  LITERAL  2330
      'T8189CSI'  LITERAL  2200
      '0'         LITERAL  1030
      25          LITERAL  1260
      40          LITERAL  1390
      5           LITERAL  1100
      8           LITERAL  2340
Indicator References:
      INDICATOR  REFERENCES (M=MODIFIED D=DEFINED)
      *IN        1030
      LR         1960M
      99         1030
      * * * * *   E N D   O F   C R O S S   R E F E R E N C E   * * * * *
                          M e s s a g e   S u m m a r y
* QRG4110 Severity: 00 Number: 1
      Message . . . . : Data-structure specified without any subfields.
* QRG7031 Severity: 00 Number: 50
      Message . . . . : The Name or indicator is not referenced.
* QRG7089 Severity: 00 Number: 1
      Message . . . . : The RPG provides Separate-Indicator area for
      file.
* QRG7111 Severity: 00 Number: 1
      Message . . . . : Unexpected source type.
      * * * * *   E N D   O F   M E S S A G E   S U M M A R Y   * * * * *
                          F i n a l   S u m m a r y
Message Count: (by Severity Number)
      TOTAL  00  10  20  30  40  50
      53    53   0   0   0   0   0
Program Source Totals:
      Records . . . . . : 468
      Specifications . . . . . : 181
      Table Records . . . . . : 0
      Comments . . . . . : 287
PRM has been called.
Program T8189CRS is placed in library APPCLIB. 00 highest Error-Severity-Code.
      * * * * *   E N D   O F   C O M P I L A T I O N   * * * * *

```

Figure F-8 (Part 10 of 10). RPG/400 Inquiry Example – Local Program

RPG/400 Remote Program for Inquiry Application (Example 3)

The following explains the structure of the RPG/400 remote program that handles requests sent by the partner program.

Program Explanation: The reference numbers in the explanation below correspond to the numbers in the program example illustrated in Figure F-9 on page F-50.

Note: On any type of error that is not expected (for example, open errors CPI Communications

return_code on a call), the session is ended and the program ends.

- 1** The file specification defines the files used in the program.

T8189DB is the database file that contains the valid part numbers and part descriptions.

All files are implicitly opened at the beginning of the RPG/400 program cycle.

The continuation lines for the file specification define the following:

KINFSR Specifies the subroutine FAIL is to be called when a file exception condition occurs.
- 2** The statement `/COPY QIRPG/QIRGINC,CMRPG` places the contents of the AS/400-supplied CPI Communications pseudonym file CMRPG in the program.
- 3** The STRCNV and GETDTA subroutines are called to start a conversation with the partner program and wait on a request by the partner program, respectively.
- 4** The program loops until there are no more requests to process, or until an error occurs in the transaction with the partner program.
- 5** A search of the database file is performed using the part number received from the partner program as the key.

Note: If the part number is not found, indicator 98 is set on.
- 6** If indicator 98 is set on (indicating that the part number was not found in the database file), the `Send_Error` (CMSERR) call is issued. As a result, a negative response to the received confirmation request is sent to the partner program.

If indicator 98 is not set on (indicating that the part number was found in the database file), the `Confirmed` (CMCFMD) call is issued. As a result, a positive response to the received confirmation request is sent to the partner program.
- 7** The `Send_Data` (CMSEND) call is issued. The data sent (set previously) is either an error message (if the part

was not found) or the part description (if the part is found). The CMSEND call is issued with the following conversation characteristics (the conversation characteristics were set in STRCNV): a *send_type* of `CM_SEND_AND_PREP_TO_RECEIVE` and a *prepare_to_receive_type* of `CM_PREP_TO_RECEIVE_FLUSH`. Setting the conversation characteristics to these values flushes the data and changes the data flow direction.

- 8** The GETDTA subroutine is called to wait on a request by the partner program.
- 9** The following section of code performs the end-of-program processing.

If the conversation is still active, it is assumed that an error was detected. The *deallocate_type* is set to `CM_DEALLOCATE_ABEND` by issuing the `Set_Deallocate_Type` (CMSDT) call, followed by a call to `Deallocate` (CMDEAL) to end the conversation abnormally.
- 10** The last record indicator (LR) is set on. All files are implicitly closed, and the program ends.
- 11** The STRCNV subroutine establishes a conversation with the partner program and sets various conversation characteristics.
- 12** The `Accept_Conversation` (CMACCP) call is issued so that a conversation can be started with the partner program.
- 13** The `Set_Prepare_To_Receive_Type` (CMSPTR) call is issued so that the *prepare_to_receive_type* conversation characteristic is set to `CM_PREP_TO_RECEIVE_FLUSH`.
- 14** The `Set_Send_Type` (CMSST) call is issued so that the *send_type* conversation characteristic is set to `CM_SEND_AND_PREP_TO_RECEIVE`.
- 15** The GETDTA subroutine waits for a request from the partner program by issuing the `Receive` (CMRCV) call.

Note: A transaction is processed if all the data is received with a turnaround indication, and the partner program

requested confirmation. This is indicated by the following:

- The value of the *data_received* variable on the CMRCV call is
CM_COMPLETE_DATA_RECEIVED
- The value of the *status_received* variable on the CMRCV call is
CM_CONFIRM_SEND_RECEIVED

16

The FAIL subroutine gets control when a file exception or error occurs. The FAIL subroutine handles all file exceptions or errors by passing control to the section of code that performs the end-of-program processing (**9**).

```

Compiler . . . . . : IBM AS/400 RPG/400
Command Options:
  Program . . . . . : APPCLIB/T8189CRT
  Source file . . . . . : QUSRTOOL/QATTRPG
  Source member . . . . . : *PGM
Text not available for message RXT0073 file QRPMSG.
  Generation options . . . . . : *NOLIST *NOXREF *NOATR *NODUMP *NOOPTIMIZE
  Source listing indentation . . . . . : *NONE
  SAA flagging . . . . . : *NOFLAG
  Generation severity level . . . . . : 29
  Print file . . . . . : *LIBL/QSYSVRT
  Replace program . . . . . : *YES
  Target release . . . . . : *CURRENT
  User profile . . . . . : *USER
  Authority . . . . . : *LIBCRTAUT
  Text . . . . . : *SRCMBRTXT
  Phase trace . . . . . : *NO
  Intermediate text dump . . . . . : *NONE
  Snap dump . . . . . : *NONE
  Codelist . . . . . : *NONE
  Ignore decimal data error . . . . . : *NO
Actual Program Source:
  Member . . . . . : T8189CRT
  File . . . . . : QATTRPG
  Library . . . . . : QUSRTOOL
  Last Change . . . . . : 09/26/90 08:27:42
SEQUENCE
NUMBER *...1....+...2....+...3....+...4....+...5....+...6....+...7...* IND DO LAST PAGE PROGRAM
USE NUM UPDATE LINE ID
Source Listing
10 H
1 20 FT8189DB IF E K DISK
30 F KINFSR FAIL
RECORD FORMAT(S): LIBRARY APPCLIB FILE T8189DB.
EXTERNAL FORMAT DBRCD RPG NAME DBRCD
40 E MSG 1 1 40
50 I*****
60 I* Program name.....: T8189CRT *
70 I* Program description.: CPI Communications remote program *
80 I* Language.....: RPG/400 *
90 I* *
100 I* This program accepts the incoming conversation by *
110 I* issuing an Accept_Conversation (CMACCP) call. It then *
120 I* issues a Receive (CMRCV) call to receive the part number *
130 I* from the remote system. When the CMRCV call completes, *
140 I* the status_received value will be CM_CONFIRM_SEND. The *
150 I* database file T8189DB is searched for the received part *
160 I* number. If the part number is found, the Confirmed *
170 I* (CMCFMD) call is issued, followed by a Send_Data *
180 I* (CMSEND) call (the data sent is the part description *
190 I* corresponding to the part number retrieved from the *
200 I* database file). However, if the part number is not *
210 I* found, the Send_Error (CMSERR) call is issued, followed *
220 I* by a CMSEND call (the data sent is a message describing *
230 I* the error). The CMSEND call sending either the part *
240 I* description or the error message is issued with a *
250 I* send_type conversation characteristic of *
260 I* CM_SEND_AND_PREP_TO_RECEIVE and a *
270 I* prepare_to_receive_type conversation characteristic of *
280 I* CM_PREP_TO_RECEIVE_FLUSH. Setting the conversation *
290 I* characteristics to these values results in the flushing *
300 I* of the data, and the changing of the data flow *
310 I* direction. The partner program can send more inquiries. *
320 I* *
330 I* This program will continue to handle inquiries from the *
340 I* partner program until a return_code that is not CM_OK *
350 I* is received. Then the program ends. *
360 I* *

```

Figure F-9 (Part 1 of 10). RPG/400 Inquiry Example – Remote Program


```

370 I* NOTE 1: If an unexpected return_code value is received on *
380 I*      any of the CPI Communications calls, the *
390 I*      program will abnormally end the conversation *
400 I*      with a deallocate_type of CM_DEALLOCATED_ABEND, *
410 I*      and program processing will end. *
420 I* *
430 I* NOTE 2: On the CMRCV call, if the data_received *
SEQUENCE NUMBER *...1....+....2....+....3....+....4....+....5....+....6....+....7...*
IND USE DO NUM LAST UPDATE PAGE LINE PROGRAM ID
440 I*      parameter value does not indicate *
450 I*      CM_COMPLETE_DATA_RECEIVED, or if the *
460 I*      status_received parameter value does not indicate *
470 I*      CM_CONFIRM_SEND_RECEIVED, the program will *
480 I*      abnormally end the conversation with a *
490 I*      deallocate_type of CM_DEALLOCATED_ABEND, *
500 I*      and program processing will end. *
510 I* *
520 I* NOTE 3: This program can be started by ANY of the *
530 I*      "local" program examples in the APPC Programmer's *
540 I*      Guide. *
550 I* *
560 I*****
570 I*
580 I*****
590 I* Use the CPI Communications supplied pseudonyms. *
600 I*****
610 I*
2 620 I/COPY QIRPG/QIRGINC,CMRPG
A000000+ MEMBER CMRPG IN FILE QIRGINC LIBRARY QIRPG OPENED FOR /COPY.
A000010+ I*
A000020+ I* RPG INCLUDE FOR SAA COMMUNICATIONS SUPPORT
A000030+ I*
B000000+ INPUT FIELDS FOR RECORD DBRCD FILE T8189DB FORMAT DBRCD.
B000001+          1 5 ITEMNH
B000002+          6 30 ITEM D
A000040+ ICMCONS DS
A000050+ I*****
A000060+ I* conversation_type values:
A000070+ I*
A000080+ I* CM_BASIC_CONVERSATION -- VALUE 0 (BASIC)
A000090+ I* CM_MAPPED_CONVERSATION -- VALUE 1 (MAPPED)
A000100+ I*
A000110+ I          0 C BASIC
A000120+ I          1 C MAPPED
A000130+ I*****
A000140+ I* data_received values:
A000150+ I*
A000160+ I* CM_NO_DATA_RECEIVED -- VALUE 0 (NODATA)
A000170+ I* CM_DATA_RECEIVED -- VALUE 1 (DATREC)
A000180+ I* CM_COMPLETE_DATA_RECEIVED -- VALUE 2 (COMDAT)
A000190+ I* CM_INCOMPLETE_DATA_RECEIVED -- VALUE 3 (INCDAT)
A000200+ I*
A000210+ I          0 C NODATA
A000220+ I          1 C DATREC
A000230+ I          2 C COMDAT
A000240+ I          3 C INCDAT
A000250+ I*****
A000260+ I* deallocate_type values:
A000270+ I*
A000280+ I* CM_DEALLOCATE_SYNC_LEVEL -- VALUE 0 (DESYNC)
A000290+ I* CM_DEALLOCATE_FLUSH -- VALUE 1 (DEFLUS)
A000300+ I* CM_DEALLOCATE_CONFIRM -- VALUE 2 (DECONF)
A000310+ I* CM_DEALLOCATE_ABEND -- VALUE 3 (DEABTY)

```

Figure F-9 (Part 2 of 10). RPG/400 Inquiry Example – Remote Program

SEQUENCE NUMBER	*...1...+...2...+...3...+...4...+...5...+...6...+...7...*	IND USE	DO NUM	LAST UPDATE	PAGE LINE	PROGRAM ID
A000320+	I*					
A000330+	I 0	C		DESYNC		
A000340+	I 1	C		DEFLUS		
A000350+	I 2	C		DECONF		
A000360+	I 3	C		DEABTY		
A000370+	I*****					
A000380+	I* error_direction values:					
A000390+	I*					
A000400+	I* CM_RECEIVE_ERROR	--	VALUE 0	(RCVERR)		
A000410+	I* CM_SEND_ERROR	--	VALUE 1	(SNDERR)		
A000420+	I*					
A000430+	I 0	C		RCVERR		
A000440+	I 1	C		SNDERR		
A000450+	I*****					
A000460+	I* fill values:					
A000470+	I*					
A000480+	I* CM_FILL_LL	--	VALUE 0	(FILLL)		
A000490+	I* CM_FILL_BUFFER	--	VALUE 1	(FILBUF)		
A000500+	I*					
A000510+	I 0	C		FILLL		
A000520+	I 1	C		FILBUF		
A000530+	I*****					
A000540+	I* prepare_to_receive_type values:					
A000550+	I*					
A000560+	I* CM_PREP_TO_RECEIVE_SYNC_LEVEL	--	VALUE 0	(PTRSL)		
A000570+	I* CM_PREP_TO_RECEIVE_FLUSH	--	VALUE 1	(PTRFLS)		
A000580+	I* CM_PREP_TO_RECEIVE_CONFIRM	--	VALUE 2	(PTRCON)		
A000590+	I*					
A000600+	I 0	C		PTRSL		
A000610+	I 1	C		PTRFLS		
A000620+	I 2	C		PTRCON		
A000630+	I*****					
A000640+	I* receive_type values:					
A000650+	I*					
A000660+	I* CM_RECEIVE_AND_WAIT	--	VALUE 0	(RCVWAT)		
A000670+	I* CM_RECEIVE_IMMEDIATE	--	VALUE 1	(RCVIMM)		
A000680+	I*					
A000690+	I 0	C		RCVWAT		
A000700+	I 1	C		RCVIMM		
A000710+	I*****					
A000720+	I* request_to_send_received values:					
A000730+	I*					
A000740+	I* CM_REQ_TO_SEND_NOT_RECEIVED	--	VALUE 0	(RTSNOT)		
A000750+	I* CM_REQ_TO_SEND_RECEIVED	--	VALUE 1	(RTSREC)		
A000760+	I*					
A000770+	I 0	C		RTSNOT		
A000780+	I 1	C		RTSREC		
A000790+	I*****					
A000800+	I* return_code values:					
A000810+	I*					
A000820+	I* CM_OK	--	VALUE 0	(CMOK)		
A000830+	I* CM_ALLOCATE_FAILURE_NO_RETRY	--	VALUE 1	(ALFLNR)		
A000840+	I* CM_ALLOCATE_FAILURE_RETRY	--	VALUE 2	(ALFLRE)		
A000850+	I* CM_CONVERSATION_TYPE_MISMATCH	--	VALUE 3	(CNVMIS)		
SEQUENCE NUMBER	*...1...+...2...+...3...+...4...+...5...+...6...+...7...*	IND USE	DO NUM	LAST UPDATE	PAGE LINE	PROGRAM ID
A000860+	I* CM_PIP_NOT_SPECIFIED_CORRECTLY	--	VALUE 5	(PIPNSC)		
A000870+	I* CM_SECURITY_NOT_VALID	--	VALUE 6	(SECNVL)		
A000880+	I* CM_SYNC_LVL_NOT_SUPPORTED_LU	--	VALUE 7	(SLNSLU)		
A000890+	I* CM_SYNC_LVL_NOT_SUPPORTED_PGH	--	VALUE 8	(SLNSP)		
A000900+	I* CM_TPN_NOT_RECOGNIZED	--	VALUE 9	(TPNAME)		
A000910+	I* CM_TP_NOT_AVAILABLE_NO_RETRY	--	VALUE 10	(TPNORE)		
A000920+	I* CM_TP_NOT_AVAILABLE_RETRY	--	VALUE 11	(TPRET)		
A000930+	I* CM_DEALLOCATED_ABEND	--	VALUE 17	(DEABND)		
A000940+	I* CM_DEALLOCATED_NORMAL	--	VALUE 18	(DENORM)		
A000950+	I* CM_PARAMETER_ERROR	--	VALUE 19	(PARERR)		
A000960+	I* CM_PRODUCT_SPECIFIC_ERROR	--	VALUE 20	(PRODER)		
A000970+	I* CM_PROGRAM_ERROR_NO_TRUNC	--	VALUE 21	(PENOTR)		
A000980+	I* CM_PROGRAM_ERROR_PURGING	--	VALUE 22	(PEPURG)		

Figure F-9 (Part 3 of 10). RPG/400 Inquiry Example – Remote Program

```

A000990+ I* CM_PROGRAM_ERROR_TRUNC -- VALUE 23 (PETRNC)
A001000+ I* CM_PROGRAM_PARAMETER_CHECK -- VALUE 24 (PEPCHK)
A001010+ I* CM_PROGRAM_STATE_CHECK -- VALUE 25 (STACHK)
A001020+ I* CM_RESOURCE_FAILURE_NO_RETRY -- VALUE 26 (RFNORE)
A001030+ I* CM_RESOURCE_FAILURE_RETRY -- VALUE 27 (RFRET)
A001040+ I* CM_UNSUCCESSFUL -- VALUE 28 (UNSUCC)
A001050+ I* CM_DEALLOCATED_ABEND_SVC -- VALUE 30 (DABSV)
A001060+ I* CM_DEALLOCATED_ABEND_TIMER -- VALUE 31 (DABTIM)
A001070+ I* CM_SVC_ERROR_NO_TRUNC -- VALUE 32 (SVCENT)
A001080+ I* CM_SVC_ERROR_PURGING -- VALUE 33 (SVCEP)
A001090+ I* CM_SVC_ERROR_TRUNC -- VALUE 34 (SVCET)
A001100+ I*
A001110+ I 0 C CMOK
A001120+ I 1 C ALFLNR
A001130+ I 2 C ALFLRE
A001140+ I 3 C CNVMIS
A001150+ I 5 C PIPNSC
A001160+ I 6 C SECNVL
A001170+ I 7 C SLNSLU
A001180+ I 8 C SLNSP
A001190+ I 9 C TPNAME
A001200+ I 10 C TPNORE
A001210+ I 11 C TPRET
A001220+ I 17 C DEABND
A001230+ I 18 C DENORM
A001240+ I 19 C PARERR
A001250+ I 20 C PRODER
A001260+ I 21 C PENOTR
A001270+ I 22 C PEPURG
A001280+ I 23 C PETRNC
A001290+ I 24 C PEPCHK
A001300+ I 25 C STACHK
A001310+ I 26 C RFNORE
A001320+ I 27 C RFRET
A001330+ I 28 C UNSUCC
A001340+ I 30 C DABSV
A001350+ I 31 C DABTIM
A001360+ I 32 C SVCENT
A001370+ I 33 C SVCEP
A001380+ I 34 C SVCET
A001390+ I*****
SEQUENCE
NUMBER *...1...+...2...+...3...+...4...+...5...+...6...+...7...* IND DO LAST PAGE PROGRAM
USE NUM UPDATE LINE ID
A001400+ I* return_control values:
A001410+ I*
A001420+ I* CM_WHEN_SESSION_ALLOCATED -- VALUE 0 (SESALL)
A001430+ I* CM_IMMEDIATE -- VALUE 1 (IMMED)
A001440+ I*
A001450+ I 0 C SESALL
A001460+ I 1 C IMMED
A001470+ I*****
A001480+ I* send_type values:
A001490+ I*
A001500+ I* CM_BUFFER_DATA -- VALUE 0 (BUFDAT)
A001510+ I* CM_SEND_AND_FLUSH -- VALUE 1 (SNDFLS)
A001520+ I* CM_SEND_AND_CONFIRM -- VALUE 2 (SNDCNF)
A001530+ I* CM_SEND_AND_PREP_TO_RECEIVE -- VALUE 3 (SNDPTR)
A001540+ I* CM_SEND_AND_DEALLOCATE -- VALUE 4 (SNDEL)
A001550+ I*
A001560+ I 0 C BUFDAT
A001570+ I 1 C SNDFLS
A001580+ I 2 C SNDCNF
A001590+ I 3 C SNDPTR
A001600+ I 4 C SNDEL
A001610+ I*****
A001620+ I* status_received values:
A001630+ I*

```

Figure F-9 (Part 4 of 10). RPG/400 Inquiry Example – Remote Program

```

A001640+ I* CM_NO_STATUS_RECEIVED -- VALUE 0 (NOSTAT)
A001650+ I* CM_SEND_RECEIVED -- VALUE 1 (SNDREC)
A001660+ I* CM_CONFIRM_RECEIVED -- VALUE 2 (CONRCV)
A001670+ I* CM_CONFIRM_SEND_RECEIVED -- VALUE 3 (CONSND)
A001680+ I* CM_CONFIRM_DEALLOC_RECEIVED -- VALUE 4 (CONDEL)
A001690+ I*
A001700+ I 0 C NOSTAT
A001710+ I 1 C SNDREC
A001720+ I 2 C CONRCV
A001730+ I 3 C CONSND
A001740+ I 4 C CONDEL
A001750+ I*****
A001760+ I* sync_level values:
A001770+ I*
A001780+ I* CM_NONE -- VALUE 0 (NONE)
A001790+ I* CM_CONFIRM -- VALUE 1 (CONFRM)
A001800+ I*
A001810+ I 0 C NONE
A001820+ I 1 C CONFRM
630 I*
* 4110 4110-**
640 ICMPARM DS
650 I 1 8 CONVID
660 I B 9 12ORTNCOD
670 I B 13 16ODATRCV
680 I B 17 20ODLCTYP
690 I B 21 24ORCVLEN
700 I B 25 28OREQLEN
710 I B 29 32OREQTSR
SEQUENCE IND DO LAST PAGE PROGRAM
NUMBER *...1....2....3....4....5....6....7...* USE NUM UPDATE LINE ID
720 I B 33 360SNDLEN
730 I B 37 400SNDTYP
740 I B 41 440STSRCV
750 I B 45 480PRRCV
760 I 49 88 SNDBUF
770 C*
780 C*****
790 C* START OF PROGRAM *
800 C* *
810 C* Files are implicitly opened, a conversation with the *
820 C* remote program is started, and the part inquiry *
830 C* processing starts. Inquiries are handled until a *
840 C* CM_DEALLOCATED_NORMAL return_code is received. *
850 C*****
860 C*
3 870 C EXSR STRCNV
880 C EXSR GETDTA
4 890 C RTNCCOD DOWEQCMOK B001
900 C*
910 C*****
920 C* A search of the database file is done using the part *
930 C* number as the key (indicator 98 is set on if the part *
940 C* number is not found). *
950 C*****
960 C*
5 970 C ITEMNM CHAINDBRCD 98 1 001
980 C*
990 C*****
1000 C* If the part number is not found, the CMSERR call is *
1010 C* issued; otherwise, the CMCFMD call is issued. *
1020 C*****

```

Figure F-9 (Part 5 of 10). RPG/400 Inquiry Example – Remote Program

```

1030 C*
6 1040 C 98          Z-ADD40      SNDLEN      001
1050 C 98          MOVELMMSG,1 SNDBUF      001
1060 C 98          CALL 'CMSERR'  001
1070 C            PARM          CONVID      001
1080 C            PARM          REQTSR      001
1090 C            PARM          RTNCOD      001
1100 C*
1110 C N98         Z-ADD25      SNDLEN      001
1120 C N98         MOVELITEMD  SNDBUF      001
1130 C N98         CALL 'CMCFMD' 001
1140 C            PARM          CONVID      001
1150 C            PARM          RTNCOD      001
1160 C            RTNCOD      CABNECMOK  ENDCNV      001
1170 C*
1180 C*****
1190 C* The CMSEND call is issued. The data sent (set previously) *
1200 C* is either an error message (if the part was not found) *
1210 C* or the part description (if the part is found). *
1220 C*****
1230 C*
SEQUENCE          IND  DO  LAST  PAGE  PROGRAM
NUMBER          *...1...+...2...+...3...+...4...+...5...+...6...+...7...*
7 1240 C          CALL 'CMSEND'  001
1250 C          PARM          CONVID      001
1260 C          PARM          SNDBUF      001
1270 C          PARM          SNDLEN      001
1280 C          PARM          REQTSR      001
1290 C          PARM          RTNCOD      001
1300 C          RTNCOD      CABNECMOK  ENDCNV      001
1310 C*
8 1320 C          EXSR GETDTA      001
1330 C          END              E001
1340 C*
1350 C*****
1360 C* The following code handles the end-of-program processing. *
1370 C* This includes the ending of the conversation with *
1380 C* the remote system (if conversation is active), and *
1390 C* the setting of the last record indicator. *
1400 C*****
1410 C*
9 1420 C          ENDCNV      TAG
1430 C*
1440 C          RTNCOD      IFNE ALFLNR      B001
1450 C          RTNCOD      ANDNEALFLRE      001
1460 C          RTNCOD      ANDNEDEABND      001
1470 C          RTNCOD      ANDNEDENORM      001
1480 C          RTNCOD      ANDNEPRODER      001
1490 C          RTNCOD      ANDNERFNORE      001
1500 C          RTNCOD      ANDNERFRET      001
1510 C          Z-ADDDEABTY  DLCTYP      001
1520 C          CALL 'CMSDT'  001
1530 C          PARM          CONVID      001
1540 C          PARM          DLCTYP      001
1550 C          PARM          RTNCOD      001
1560 C          CALL 'CMDEAL'  001
1570 C          PARM          CONVID      001
1580 C          PARM          RTNCOD      001
1590 C          END              E001
1600 C*
1610 C          ENDPGM      TAG
10 1620 C          SETON          LR          1
1630 C*
1640 C*****
1650 C* "STRCNV" subroutine. *
1660 C* *
1670 C* This subroutine establishes a conversation with the *
1680 C* remote program, and also sets various conversation *
1690 C* characteristics. *
1700 C*****

```

Figure F-9 (Part 6 of 10). RPG/400 Inquiry Example – Remote Program

```

1710 C*
11 1720 CSR          STRCNV  BEGSR
1730 C*
1740 C*****
1750 C* The CMACCP call is issued so that a conversation can be *
1760 C* started with the partner program. *
1770 C*****
SEQUENCE          IND  DO  LAST  PAGE  PROGRAM
NUMBER          *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE  NUM  UPDATE  LINE  ID
1780 C*
12 1790 CSR          CALL 'CMACCP'
1800 CSR          PARM          CONVID
1810 CSR          PARM          RTNCOD
1820 CSR          RTNCOD  CABNECMOK  ENDPGM
1830 C*
1840 C*****
1850 C* The Set_Prepare_To_Receive_Type (CMSPTR) call is issued *
1860 C* so that the prepare_to_receive_type conversation *
1870 C* characteristic is set to CM_PREP_TO_RECEIVE_FLUSH. *
1880 C*****
1890 C*
13 1900 CSR          Z-ADDPTRFLS  PRERCV
1910 CSR          CALL 'CMSPTR'
1920 CSR          PARM          CONVID
1930 CSR          PARM          PRERCV
1940 CSR          PARM          RTNCOD
1950 C*
1960 C*****
1970 C* The Set_Send_Type (CMSST) call is issued so that the *
1980 C* send_type conversation characteristic is set to *
1990 C* CM_SEND_AND_PREP_TO_RECEIVE. *
2000 C*****
2010 C*
2020 CSR          Z-ADDSNDPTR  SNDTYP
14 2030 CSR          CALL 'CMSST'
2040 CSR          PARM          CONVID
2050 CSR          PARM          SNDTYP
2060 CSR          PARM          RTNCOD
2070 C*
2080 C*
2090 CSR          ENDSR
2100 C*
2110 C*****
2120 C* "GETDTA" subroutine. *
2130 C* *
2140 C* This subroutine waits for incoming data from the partner *
2150 C* program by issuing the CMRCV call. *
2160 C*****
2170 C*
15 2180 CSR          GETDTA  BEGSR
2190 CSR          Z-ADD5    REQLEN
2200 CSR          CALL 'CMRCV'
2210 CSR          PARM          CONVID
2220 CSR          PARM          ITEMNH
2230 CSR          PARM          REQLEN
2240 CSR          PARM          DATRCV
2250 CSR          PARM          RCVLEN
2260 CSR          PARM          STSRCV
2270 CSR          PARM          REQTSR
2280 CSR          PARM          RTNCOD
2290 C*
2300 CSR          RTNCOD  IFEQ CMOK
2310 CSR          DATRCV  IFNE COMDAT
SEQUENCE          IND  DO  LAST  PAGE  PROGRAM
NUMBER          *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE  NUM  UPDATE  LINE  ID
2320 CSR          STSRCV  ORNE CONSND
2330 CSR          GOTO ENDCNV
2340 CSR          END
2350 CSR          END
2360 CSR          ENDSR
2370 C*

```

Figure F-9 (Part 7 of 10). RPG/400 Inquiry Example – Remote Program

```

2380 C*****
2390 C* "FAIL" subroutine.
2400 C*
2410 C* This subroutine handles file exception/errors.
2420 C*****
2430 C*
16 2440 CSR          FAIL          BEGSR
2450 CSR          GOTO ENDCNV
2460 CSR          ENDSR
      * * * * * E N D   O F   S O U R C E   * * * * *
      A d d i t i o n a l   D i a g n o s t i c   M e s s a g e s
* 7111 SOURCE FILE MEMBER HAS AN UNEXPECTED SOURCE TYPE.
SEQUENCE
NUMBER *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
      C o m p i l e - T i m e   T a b l e s
Table/Array . . . . . : MSG
2480 The requested part was not found.
      K e y   F i e l d   I n f o r m a t i o n
      P H Y S I C A L   L O G I C A L
      F I L E / R C D   F I E L D   F I E L D   A T T R I B U T E S
01 T8189DB
      D B R C D
      I T E M N M           C H A R   5
      C r o s s   R e f e r e n c e
File and Record References:
      F I L E / R C D   D E V / R C D   R E F E R E N C E S   ( D = D E F I N E D )
01 T8189DB   DISK           20D
      D B R C D           20D B000000   970
Field References:
      F I E L D   A T T R   R E F E R E N C E S   ( M = M O D I F I E D   D = D E F I N E D )
      A L F L N R   C O N S T   A 0 0 1 1 2 0 D   1 4 4 0
      A L F L R E   C O N S T   A 0 0 1 1 3 0 D   1 4 5 0
* 7031 BASIC      C O N S T   A 0 0 0 1 1 0 D
* 7031 BUFDAT     C O N S T   A 0 0 1 5 6 0 D
* 7031 CMCONS    D S ( 1 )   A 0 0 0 0 4 0 D
      C M O K       C O N S T   A 0 0 1 1 1 0 D   8 9 0   1 1 6 0   1 3 0 0   1 8 2 0
      2 3 0 0
* 7031 CMPARM    D S ( 8 8 )   6 4 0 D
* 7031 CNVMIS    C O N S T   A 0 0 1 1 4 0 D
      C O M D A T     C O N S T   A 0 0 0 2 3 0 D   2 3 1 0
* 7031 CONDEL    C O N S T   A 0 0 1 7 4 0 D
* 7031 CONFRM    C O N S T   A 0 0 1 8 2 0 D
* 7031 CONRCV    C O N S T   A 0 0 1 7 2 0 D
      C O N S N D     C O N S T   A 0 0 1 7 3 0 D   2 3 2 0
      C O N V I D     A ( 8 )     6 5 0 D   1 0 7 0   1 1 4 0   1 2 5 0   1 5 3 0
      1 5 7 0   1 8 0 0   1 9 2 0   2 0 4 0   2 2 1 0
* 7031 DABSVC    C O N S T   A 0 0 1 3 4 0 D
* 7031 DABTIM    C O N S T   A 0 0 1 3 5 0 D
      D A T R C V     B ( 9 , 0 )   6 7 0 D   2 2 4 0   2 3 1 0
* 7031 DATREC    C O N S T   A 0 0 0 2 2 0 D
      D E A B N D     C O N S T   A 0 0 1 2 2 0 D   1 4 6 0
      D E A B T Y     C O N S T   A 0 0 0 3 6 0 D   1 5 1 0
* 7031 DECONF    C O N S T   A 0 0 0 3 5 0 D
* 7031 DEFLUS    C O N S T   A 0 0 0 3 4 0 D
      D E N O R M     C O N S T   A 0 0 1 2 3 0 D   1 4 7 0
* 7031 DESYNC    C O N S T   A 0 0 0 3 3 0 D
      D L C T Y P     B ( 9 , 0 )   6 8 0 D   1 5 1 0 M   1 5 4 0
      E N D C N V     T A G           1 1 6 0   1 3 0 0   1 4 2 0 D   2 3 3 0   2 4 5 0
      E N D P G H     T A G           1 6 1 0 D   1 8 2 0
      F A I L         B E G S R       2 0   2 4 4 0 D
* 7031 FILBUF    C O N S T   A 0 0 0 5 2 0 D
* 7031 FILL      C O N S T   A 0 0 0 5 1 0 D
      G E T D T A     B E G S R       8 8 0   1 3 2 0   2 1 8 0 D
* 7031 IMMED     C O N S T   A 0 0 1 4 6 0 D
* 7031 INCDAT    C O N S T   A 0 0 0 2 4 0 D
      I T E M D       A ( 2 5 )     B 0 0 0 0 0 2 D   1 1 2 0
      I T E M N M     A ( 5 )       B 0 0 0 0 0 1 D   9 7 0   2 2 2 0
* 7031 MAPPED    C O N S T   A 0 0 0 1 2 0 D

```

Figure F-9 (Part 8 of 10). RPG/400 Inquiry Example — Remote Program

	MSG(1)	A(40)	40D					
	MSG,1		1050					
* 7031	NODATA	CONST	A000210D					
* 7031	NONE	CONST	A001810D					
* 7031	NOSTAT	CONST	A001700D					
* 7031	PARERR	CONST	A001240D					
* 7031	PENOTR	CONST	A001260D					
* 7031	PEPCHK	CONST	A001290D					
* 7031	PEPURG	CONST	A001270D					
* 7031	PETRNC	CONST	A001280D					
* 7031	PIPNSC	CONST	A001150D					
	PRRCV	B(9,0)	750D	1900M	1930			
	PRODER	CONST	A001250D	1480				
* 7031	PTRCON	CONST	A000620D					
	PTRFLS	CONST	A000610D	1900				
* 7031	PTRSL	CONST	A000600D					
* 7031	RCVERR	CONST	A000430D					
* 7031	RCVIMM	CONST	A000700D					
	RCVLEN	B(9,0)	690D	2250				
* 7031	RCVWAT	CONST	A000690D					
	REQLEN	B(9,0)	700D	2190M	2230			
	REQTSR	B(9,0)	710D	1080	1280	2270		
	RFNORE	CONST	A001310D	1490				
	RFRET	CONST	A001320D	1500				
	RTNCOD	B(9,0)	660D	890	1090	1150	1160	
			1290	1300	1440	1450	1460	
			1470	1480	1490	1500	1550	
			1580	1810	1820	1940	2060	
			2280	2300				
* 7031	RTSNOT	CONST	A000770D					
* 7031	RTSREC	CONST	A000780D					
* 7031	SECNVL	CONST	A001160D					
* 7031	SESALL	CONST	A001450D					
* 7031	SLNSLU	CONST	A001170D					
* 7031	SLNSP	CONST	A001180D					
	SNDBUF	A(40)	760D	1050M	1120M	1260		
* 7031	SNDCNF	CONST	A001580D					
* 7031	SNDDEL	CONST	A001600D					
* 7031	SNDERR	CONST	A000440D					
* 7031	SNDFLS	CONST	A001570D					
	SNDLEN	B(9,0)	720D	1040M	1110M	1270		
	SNDPTR	CONST	A001590D	2020				
* 7031	SNDREC	CONST	A001710D					
	SNDTYP	B(9,0)	730D	2020M	2050			
* 7031	STACHK	CONST	A001300D					
	STRCNV	BEGSR	870	1720D				
	STRRCV	B(9,0)	740D	2260	2320			
* 7031	SVCENT	CONST	A001360D					
* 7031	SVCEP	CONST	A001370D					
* 7031	SVCET	CONST	A001380D					
* 7031	TPNAME	CONST	A001190D					
* 7031	TPNORE	CONST	A001200D					
* 7031	TPRET	CONST	A001210D					
* 7031	UNSUCC	CONST	A001330D					
	'CMACCP'	LITERAL	1790					
	'CMCFMD'	LITERAL	1130					
	'CMDEAL'	LITERAL	1560					
	'CMRCV'	LITERAL	2200					
	'CMSDT'	LITERAL	1520					
	'CMSEND'	LITERAL	1240					
	'CMSERR'	LITERAL	1060					
	'CMSPTR'	LITERAL	1910					
	'CMSST'	LITERAL	2030					
	1	LITERAL	1050					
	25	LITERAL	1110					
	40	LITERAL	1040					
	5	LITERAL	2190					

Figure F-9 (Part 9 of 10). RPG/400 Inquiry Example – Remote Program


```

Indicator References:
  INDICATOR REFERENCES (M=MODIFIED D=DEFINED)
  LR          1620M
  98          970M  1040  1050  1060  1110  1120
            1130
  * * * * *   E N D   O F   C R O S S   R E F E R E N C E   * * * * *
                M e s s a g e   S u m m a r y
* QRG4110 Severity: 00 Number: 1
  Message . . . : Data-structure specified without any subfields.
* QRG7031 Severity: 00 Number: 52
  Message . . . : The Name or indicator is not referenced.
* QRG7111 Severity: 00 Number: 1
  Message . . . : Unexpected source type.
  * * * * *   E N D   O F   M E S S A G E   S U M M A R Y   * * * * *
                F i n a l   S u m m a r y
Message Count: (by Severity Number)
  TOTAL  00  10  20  30  40  50
        54  54  0  0  0  0  0
Program Source Totals:
  Records . . . . . : 430
  Specifications . . . . . : 163
  Table Records . . . . . : 1
  Comments . . . . . : 265
PRM has been called.
Program T8189CRT is placed in library APPCLIB. 00 highest Error-Severity-Code.
  * * * * *   E N D   O F   C O M P I L A T I O N   * * * * *

```

Figure F-9 (Part 10 of 10). RPG/400 Inquiry Example — Remote Program

Bibliography

This section lists a subset of IBM publications that contain information about topics described or referred to in this guide. Refer to the *Publications Guide*, GC41-9678, if you want a list of all manuals in the AS/400 library, and a description of the information contained in those manuals. The manuals in this section are listed with their full title and base order number, but when referred to in text, a shortened version of the title is used.

Planning and Installation Manuals

The following planning and installation manuals contain information you may need when you use the AS/400 APPC support.

- *Device Configuration Guide*, SC41-8106. **Short title:** *Device Configuration Guide*.
- *Network Planning Guide*, GC41-9861. **Short title:** *Network Planning Guide*.

Customer and System Operation Manuals

The following customer and system operation manuals contain information you may need when you use the AS/400 APPC support.

- *Programming: Work Management Guide*, SC41-8078. **Short title:** *Work Management Guide*.
- *Publications Guide*, GC41-9678. **Short title:** *Publications Guide*.

AS/400 Communications Manuals

The following AS/400 communications manuals contain information you may need when you use the AS/400 APPC support.

- *Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188. **Short title:** *APPN Guide*.
- *CICS/400 Administration and Operations Guide*, SC33-0821. **Short title:** *CICS/400 Administration and Operations Guide*.
- *CICS/400 Application Programming Guide*, SC33-0822. **Short title:** *CICS/400 Application Programming Guide*.
- *Communications: Distribution Services Network Guide*, SC41-9588. **Short title:** *Distribution Services Network Guide*.
- *Communications: Integrated Services Digital Network Guide*, SC41-0003. **Short title:** *ISDN Guide*.

- *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590. **Short title:** *ICF Programmer's Guide*.
- *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001. **Short title:** *OS/400* Communications Configuration Reference*.
- *Communications and Systems Management Guide (Alerts and Distributed Systems Node Executive)*, SC41-9661. **Short title:** *Alerts and DSNX Guide*.
- *Communications: Remote Work Station Guide*, SC41-0002. **Short title:** *Remote Work Station Guide*.
- *Communications: 3270 Device Emulation Guide*, SC41-9602. **Short title:** *3270 Device Emulation Guide*.

AS/400 Programming Manuals

The following AS/400 programming manuals contain information you may need when you use the AS/400 APPC support.

- *Data Description Specifications Reference*, SC41-9620. **Short title:** *DDS Reference*.
- *Distributed Data Management Guide*, SC41-9600. **Short title:** *DDM Guide*.
- *Programming: Control Language Programmer's Guide*, SC41-8077. **Short title:** *CL Programmer's Guide*.
- *Programming: Control Language Reference*, SC41-0030. **Short title:** *CL Reference*.
- *Security Reference*, SC41-8083. **Short title:** *Security Reference*.

PC Support/400 Manuals

- *PC Support/400: DOS Installation and Administration Guide*, SC41-0006. **Short title:** *PC Support/400 DOS Installation and Administration Guide*.
- *PC Support/400: OS/2 Installation and Administration Guide*, SC41-0007. **Short title:** *PC Support/400 OS/2 Installation and Administration Guide*.

Communications Architectures

The following manuals provide information on the communications protocols: SNA via synchronous data link control (SDLC), X.25 packet-switched data networks, ISDN networks, and IBM Token-Ring Networks. Refer to these manuals for descriptions of communications protocols.

- *SNA Formats*, GA27-3136.
- *SNA Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*, SC30-3269, TNL: SN30-3562.
- *SNA Transaction Programmer's Reference Manual for LU Type 6.2*, GC30-3084, TNL: GN61-0101.

SAA CPI Communications

Systems Application Architecture: Common Programming Interface Communications Reference, SC26-4399.

Index

A

- acquire operation 5-4
- acquire-program-device operation E-4, E-21, E-37
- ACTIVATE_SESSION session control verb C-2
- Add Communications Entry (ADDCMNE) command
 - DFTUSR parameter 3-15
- Add Intersystem Communications Function Program Device Entry (ADDICFDEVE) command
 - description 5-2
 - parameters 5-2
- ADDCMNE (Add Communications Entry) command
 - DFTUSR parameter 3-15
- ADDICFDEVE (Add Intersystem Communications Function Program Device Entry) command
 - description 5-2
 - parameters 5-2
- advanced peer-to-peer networking (APPN)
 - configuration
 - See *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001
 - definition 1-1
 - network
 - See also *Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188
 - configuring 2-4
 - low-entry networking node limitations 2-4
 - remote location names 3-4
 - support
 - relationship to APPC support 1-1
- advanced program-to-program communications (APPC) (continued)
 - application considerations
 - ICF 7-1
 - basic 3-2, 3-3
 - configuration
 - See also *Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188
 - configuring an ISDN network 2-1
 - defining 2-1
 - in an ISDN network 2-1
 - list of commands used 2-1
 - network (without APPN) 2-4
 - configuration examples D-1
 - configuring 2-1
 - conversation 3-2, 3-3
 - asynchronous 3-1
 - basic 3-2
 - limitations 1-2
 - mapped 3-2
 - synchronous 3-1
 - definition 1-1, 2-1
 - description and capabilities 1-1
 - devices created by the system 3-13
 - entering commands 2-1
 - ICF file operations
 - chart A-1
 - entering commands 2-1
 - ICF file operations
 - chart A-1
 - invalid password attempts 3-12
 - line description 2-1
 - mapped 3-2
 - network security 3-7
 - relationship to APPN support 1-1
 - remote location names 3-4, 3-5
 - remote systems supported 1-1
 - return code
 - detailed descriptions B-4
 - running 4-1
 - security considerations 3-6
 - session description 3-1
 - support provided by 1-1
 - unit-of-work identifier 3-6
 - without APPN 3-4
 - writing application programs
 - CPI Communications 6-1
- alert
 - definition 4-9
- alert support
 - definition 1-2
- allow-write (ALWVRT) function 5-9
- APPC (advanced program-to-program communications)
 - application considerations
 - ICF 7-1
 - basic 3-2, 3-3
 - configuration
 - See also *Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188
 - configuring an ISDN network 2-1
 - defining 2-1
 - in an ISDN network 2-1
 - list of commands used 2-1
 - network (without APPN) 2-4
 - configuration examples D-1
 - configuring 2-1
 - conversation 3-2, 3-3
 - asynchronous 3-1
 - basic 3-2
 - limitations 1-2
 - mapped 3-2
 - synchronous 3-1
 - definition 1-1, 2-1
 - description and capabilities 1-1
 - devices created by the system 3-13
 - entering commands 2-1
 - ICF file operations
 - chart A-1

APPC (advanced program-to-program communications) (continued)

- invalid password attempts 3-12
- line description 2-1
- mapped 3-2
- network security 3-7
- relationship to APPN support 1-1
- remote location names 3-4, 3-5
- remote systems supported 1-1
- return code
 - detailed descriptions B-4
- running 4-1
- security considerations 3-6
- session description 3-1
- support provided by 1-1
- unit-of-work identifier 3-6
- without APPN 3-4
- writing application programs
 - CPI Communications 6-1

application considerations

- CPI Communications
 - Accept_Conversation call 8-1
 - Allocate call 8-1
 - multiple conversations 8-1
 - performance 8-1
 - prestart job considerations 8-2
 - Trace CPI Communications (TRCCPIC) command 8-2

ICF

- close operation 7-4
- confirm (CONFIRM) function 7-3
- end-of-session (EOS) function 7-4
- general considerations 7-1
- input 7-2
- open or acquire operation 7-1
- output 7-2
- prestart jobs 7-4
- release operation 7-4
- WAITFILE parameter 7-2

application program

- IBM-supplied 1-2
- interface 1-1
- user-written
 - using CPI Communications calls 1-2
 - using EXEC CICS commands 1-1
 - using ICF files 1-1

application program security

- CPI Communications
 - description 3-15
- ICF
 - DDS security keyword 3-13
 - description 3-13
 - keyword, DDS security 3-13
 - remote program initiates transaction 3-15
 - user program initiates transaction 3-14
 - system-supplied formats security 3-15

APPN (advanced peer-to-peer networking)

- configuration
 - See *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001

APPN (advanced peer-to-peer networking) (continued)

- definition 1-1
- network
 - See also *Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188
 - configuring 2-4
 - low-entry networking node limitations 2-4
 - remote location names 3-4
 - support
 - relationship to APPC support 1-1

architected length (LL)

- definition 3-2
- description 3-3

architecture

- LU type 6.2 1-1
- node type 2.1 1-1

AS/400 system

- remote systems
 - APPC support 1-1

AS/400 system manuals ix, G-1

asynchronous conversation

- definition 3-1

attribute information

- GET_ATTRIBUTES basic conversation verb table C-14

authority

- security officer 3-13

B

basic conversation

- ICF 3-3
- required knowledge 3-2

basic conversation verb

- CONFIRMED C-13
- FLUSH C-14
- GET_ATTRIBUTES C-14
- LU type 6.2 architecture C-12
- POST_ON_RECEIPT C-16
- PREPARE_TO_RECEIVE C-16
- RECEIVE_AND_WAIT C-17
- RECEIVE_IMMEDIATE C-18
- REQUEST_TO_SEND C-18
- SEND_DATA C-18
- SEND_ERROR C-19
- TEST C-19

C

C/400 programming language

- example program
 - CPI Communications local program for inquiry applications F-2
 - CPI Communications remote program for inquiry applications F-11
 - ICF local program for inquiry applications E-3
 - ICF remote program for inquiry applications E-12
- user-written applications 1-1

call
 CPI Communications 6-5

Change Communications Side Information (CHGCSI) command
 parameters 6-2

Change Intersystem Communications Function File (CHGICFF) command
 description 5-1

Change Intersystem Communications Function Program Device Entry (CHGICFDEVE) command
 description 5-2
 parameters 5-2

Change Session Maximum (CHGSSNMAX) command
 description 4-4
 parameters 4-5

change-number-of-sessions verb
 control operator verbs C-1

CHANGE_SESSION_LIMIT control operator verb C-1

characteristics of CPI Communications
See Systems Application Architecture Common Programming Interface Communications Reference, SC26-4399*

CHGCSI (Change Communications Side Information) command
 parameters 6-2

CHGICFDEVE (Change Intersystem Communications Function Program Device Entry) command
 description 5-2
 parameters 5-2

CHGICFF (Change Intersystem Communications Function File) command 5-1

CHGSSNMAX (Change Session Maximum) command
 description 4-4
 parameters 4-5

CICS/400 support
See also CICS/400 Administration and Operations Guide, SC33-0821
 definition 1-2
 writing applications 1-1

CICS/VS (Customer Information Control System for Virtual Storage)
 requirement for APPC support 1-1

class of service 6-2

close operation
 description 5-11
 ICF general considerations 7-4

COBOL/400 programming language
 example program
 CPI Communications local program for inquiry applications F-18
 CPI Communications remote program for inquiry applications F-27
 ICF local program for inquiry applications E-21
 ICF remote program for inquiry applications E-29
 user-written applications 1-1

command prompt 2-1

command, CL
See also Communications: Management Guide, SC41-0024
 Add Intersystem Communications Function Program Device Entry (ADDICFDEVE) command 5-2
 Change Communications Side Information (CHGCSI) 6-2
 Change Intersystem Communications Function File (CHGICFF) command 5-1
 Change Intersystem Communications Function Program Device Entry (CHGICFDEVE) command 5-2
 Change Session Maximum (CHGSSNMAX) 4-4
 communications side information 6-2
 Create Communications Side Information (CRTCSI) 6-2
 Create Intersystem Communications Function File (CRTICFF) command 5-1
 Delete Communications Side Information (DLTCSI) 6-2
 Delete File (DLTF) command 5-1
 Display Communications Side Information (DSPCSI) 6-2
 Display File Description (DSPFD) command 5-1
 Display File Field Description (DSPFFD) command 5-2
 Display Model Status (DSPMODSTS) 4-6
 End Mode (ENDMOD) 4-3
 Override Intersystem Communications Function File (OVRICFF) command 5-1
 Override Intersystem Communications Function Program Device Entry (OVRICFDEVE) command 5-2
 Remove Intersystem Communications Function Program Device Entry (RMVICFDEVE) command 5-2
 Start Mode (STRMOD) 4-2
 Trace CPI Communications (TRCCPIC) command 8-2
 Vary Configuration (VRYCFG) 4-1
 Work with Communications Side Information (WRKCSI) 6-2

communications
See also Communications: Operating System/400 Communications Configuration Reference, SC41-0001*
 architecture manuals G-1
 configuration
 menu driven 1-3
 data stream 3-2
 lines supported 1-3

communications entry
See also Programming: Work Management Guide, SC41-8078
 defining 3-16

communications equipment
See Network Planning Guide, GC41-9861

- communications side information** 6-1
 - commands 6-2
 - CPI Communications counterpart 6-1
 - description 6-1
 - managing 6-2
 - specifying parameter values 6-2
- communications side information object T8189CSI example F-1**
- configuration**
 - See also *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001
 - APPC network without APPN 2-4
 - communications
 - menu driven 1-3
 - remote location names 3-4
 - requirements 1-3
 - specifying with APPN (*NO) 3-4
 - specifying with APPN (*YES) 3-5
 - configuration description**
 - controller 2-2
 - deleting 2-4
 - device 2-2
 - line 2-1
 - mode 2-3
 - configuration example D-1**
 - configuring**
 - APPC 2-1
 - APPN network
 - See *Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188
 - confirm (CONFIRM) function**
 - description 5-6
 - ICF general considerations 7-3
 - confirmation**
 - effect on performance 8-2
 - CONFIRMED basic conversation verb C-13**
 - connection list**
 - definition 2-1
 - control data (CTLDTA) function**
 - description 5-6
 - keyword 5-6
 - control operator verb**
 - CHANGE_SESSION_LIMIT C-1
 - INITIALIZE_SESSION_LIMIT C-1
 - PROCESS_SESSION_LIMIT C-1
 - RESET_SESSION_LIMIT C-2
 - controller**
 - limitations 2-3
 - naming devices 2-3
 - controller description**
 - creating
 - nonswitched connection example D-4
 - switched connection example D-2
 - definition 2-2
 - for ISDN networks 2-2
 - for programs communicating on the same system 2-2
 - controller description (continued)**
 - for use with SNA pass-through 2-2
 - X.21 short-hold mode example D-8
 - controlling**
 - mode 4-2
 - conversation**
 - APPC 3-1, 3-2, 3-3
 - asynchronous 3-1
 - basic 3-2, 3-3
 - CPI Communications 3-2
 - definition 3-2
 - ICF 3-2
 - mapped 3-2
 - synchronous 3-1
 - transaction 3-2
 - conversation verb**
 - basic C-12
 - option sets C-23
 - CPI Communications**
 - See also *Systems Application Architecture* Common Programming Interface Communications Reference*, SC26-4399
 - application program security 3-15
 - characteristics 6-4
 - communications side information T8189CSI
 - example F-1
 - conversation 3-2
 - corresponding ICF operation or function
 - table 6-7
 - database file T8189DB example F-2
 - display file T8189DSP example F-2
 - files for defining pseudonyms 6-6
 - flow diagram for inquiry application example 6-11
 - mapped 3-2
 - mapping to LU 6.2 return codes 6-14
 - program calls
 - characteristics 6-4
 - description 6-4
 - input and output parameters 6-4
 - lists 6-5
 - supported by AS/400 system 6-5
 - return codes B-45
 - writing APPC application programs 6-1
 - writing application programs 1-2
 - Create Communications Side Information (CRTCSI) command**
 - parameters 6-2
 - Create Intersystem Communications Function File (CRTICFF) command**
 - description 5-1
 - creating**
 - controller description
 - nonswitched connection example D-4
 - switched connection example D-2
 - X.21 short-hold mode example D-7, D-8
 - device description
 - nonswitched connection example D-5
 - switched connection example D-2
 - X.21 short-hold mode example D-8, D-9

creating (*continued*)
line description
 nonswitched connection example D-4
 switched connection example D-1
 X.21 short-hold mode example D-6, D-8
CRTCSI (Create Communications Side Information)
command
 parameters 6-2
CRTICFF (Create Intersystem Communications Function File) command 5-1

D

data network identification code (DNIC) D-7
data queue
 waiting to use 5-8
data stream
communications
 definition 3-2
 general
 definition 3-2
database file object T8189DB example E-3, F-2
DDM (distributed data management)
 See also Distributed Data Management Guide, SC41-9600
 definition 1-2
DDS keyword
 ALWWRT keyword A-2
 CONFIRM keyword A-2
 CTLDTA keyword A-2
 DETACH keyword A-2
 DFREVOKE keyword A-2
 EOS keyword A-2
 EVOKE keyword A-2
 FAIL keyword A-2
 FMTNAME keyword A-2
 FRCDTA keyword A-2
 INVITE keyword A-2
 RCVCONFIRM keyword A-2
 RCVCTLDTA keyword A-2
 RCVDETACH keyword A-2
 RCVFAIL keyword A-2
 RCVTRNRND keyword A-2
 RECID keyword A-2
 RQSWRT keyword A-2
 RSPCONFIRM keyword A-2
 SECURITY keyword 3-13, A-2
 SYNLVL keyword A-2
 TIMER keyword A-2
 VARLEN keyword A-2
DEACTIVATE_SESSION session control verb C-2
DEFINE_LOCAL_LU definition verb C-2
DEFINE_MODE definition verb C-2
DEFINE_REMOTE_LU definition verb C-2
Delete File (DLTF) command
 description 5-1
deleting
 configuration descriptions 2-4

detach function 5-10
device
 deciding which to use 3-4
 system-created for APPC 3-13
device description
 definition 2-2
 nonswitched connection example D-5
 switched connection example D-2
 X.21 short-hold mode example D-8, D-9
display file
 waiting to use 5-8
Display File Description (DSPFD) command
 description 5-1
Display File Field Description (DSPFFD) command
 description 5-2
display file object T8189DSP example E-2, F-2
Display Mode Status (DSPMODSTS) command
 description 4-6
 parameters 4-6
display station pass-through
 See also Communications: Remote Work Station Guide, SC41-0002
 definition 1-2
DISPLAY_LOCAL_LU definition verb C-3
DISPLAY_MODE definition verb C-3
DISPLAY_REMOTE_LU definition verb C-3
distributed data management (DDM)
 See also Distributed Data Management Guide, SC41-9600
 definition 1-2
distribution services
 See also Communications: Distribution Services Network Guide, SC41-9588
 SNA (Systems Network Architecture) 1-2
DLTF (Delete File) command 5-1
DNIC (data network identification code) D-7
DSPFD (Display File Description) command 5-1
DSPFFD (Display File Field Description) command 5-2
DSPMODSTS (Display Mode Status) command
 description 4-6
duplicate profiles 3-12
 See also Security Reference, SC41-8083

E

electronic customer support
 See also Device Configuration Guide, SC41-8106
 definition 1-2
End Mode (ENDMOD) command
 parameters 4-3
end-of-session (EOS) function
 description 5-10
 ICF general considerations 7-4
 sending APPC FMH7 5-10
ending
 communications support 4-1
ENDMOD (End Mode) command
 parameters 4-3

error

program start request B-1

Ethernet

communications line used by APPC 1-3

evoked function

description 5-4

sending APPC FMH5 5-4

using device configured as SNGSSN(*YES) 5-4

example

configuration

defining controller descriptions for programs

communicating on the same system D-9

nonswitched connection without APPN D-3

switched connection without APPN D-1

two-system APPC network D-1, D-3

X.21 short-hold mode D-6

objects used by CPI Communications

communications side information F-1

database file F-2

display file F-2

objects used by ICF programs

database file object E-3

display file object E-2

ICF file object E-1

program

C/400 CPI Communications local program for inquiry application F-2

C/400 CPI Communications remote program for inquiry application F-11

C/400 ICF local program for inquiry application E-3

C/400 ICF remote program for inquiry application E-12

COBOL/400 CPI Communications local program for inquiry application F-18

COBOL/400 CPI Communications remote program for inquiry application F-27

COBOL/400 ICF local program for inquiry application E-21

COBOL/400 ICF remote program for inquiry application E-29

RPG/400 CPI Communications local program for inquiry application F-35

RPG/400 CPI Communications remote program for inquiry application F-47

RPG/400 ICF local program for inquiry application E-36

RPG/400 ICF remote program for inquiry application E-44

using CPI Communications F-1

using ICF E-1

F

fail function

sending APPC FMH7 5-9

feedback area E-4

file object example E-1, E-2, E-3

file transfer support (FTS)

See also *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590

definition 1-2

flip-flop

half-duplex 3-1

send-and-receive mode 3-1

flow diagram

CPI Communications inquiry application 6-11

general description 5-18

ICF inquiry applications 5-19

inquiry applications using LU 6.2 verbs 5-20

FLUSH basic conversation verb C-14

force-data (FRCDTA) function 5-6

format-name (FMTNAME) function 5-6

FORTRAN/400 programming language

user-written applications 1-1

FTS (file transfer support)

See also *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590

definition 1-2

function

allow-write (ALWWRT) 5-9

CONFIRM 5-6

control data (CTLDTA) 5-6

DETACH 5-10

end-of-session (EOS) 5-10

FAIL 5-9

force-data (FRCDTA) 5-6

format-name (FMTNAME) 5-6

INVITE 5-7

request-to-write (RQSWRT) 5-9

respond-to-confirm (RSPCONFIRM) 5-9

timer 5-10

variable buffer management (VARBUFMGMT)

on read operations 5-7

on write operations 5-6

G

GDS ID (general data stream identifier)

definition 3-2

general data stream identifier (GDS ID)

definition 3-2

general information

AS/400 system manuals G-1

communications architecture manuals G-1

customer manuals G-1

planning and installation manuals G-1

programming manuals G-1

system operation manuals G-1

general security considerations 3-11

get-attributes operation 5-10

GET_ATTRIBUTES basic conversation verb C-14

GET_TP_PROPERTIES verb C-19

GET_TYPE verb C-20

H

half-duplex flip-flop protocol 3-1

high-level language (HLL)

chart A-1

HLL (high-level language)

chart A-1

I

I/O feedback area 5-12

See also Communications: Intersystem Communications Function Programmer's Guide, SC41-9590

IBM-supplied applications 1-2

ICF (intersystem communications function)

application program security 3-13

conversation

basic 3-3

corresponding CPI Communications calls

table 6-7

file E-1

See also Communications: Intersystem Communications Function Programmer's Guide, SC41-9590

commands for managing ICF file 5-1

description 5-1

object example E-1

waiting to use 5-8

writing application programs 1-1

interface

application considerations 7-1

language operations

chart A-1

operation

close 5-11

get-attributes 5-10

read 5-7

read-from-invited-program-devices 5-7

release 5-10

program

example of objects used E-1

file object example E-1, E-2, E-3

return code 0014

turnaround indication E-13

IDLC (integrated services digital network data link control)

communications line used by APPC 1-3

increasing performance

buffer size 8-1

INITIALIZE_SESSION_LIMIT control operator

verb C-1

input

ICF general considerations 7-2

input parameter

description 6-4

inquiry application

CPI Communications local program example

C/400 F-2

COBOL/400 F-18

inquiry application (continued)

CPI Communications local program example (*continued*)

RPG/400 F-35

CPI Communications remote program example

C/400 F-11

COBOL/400 F-27

RPG/400 F-47

ICF local program example

C/400 E-3

COBOL/400 E-21

RPG/400 E-36

ICF remote program example

C/400 E-12

COBOL/400 E-29

RPG/400 E-44

integrated services digital network (ISDN)

connection list 2-1

definition 1-3

network interface description 2-1

integrated services digital network data link control (IDLC)

communications line used by APPC 1-3

interface

application program 1-1

intersystem communications function (ICF)

application program security 3-13

conversation

basic 3-3

corresponding CPI Communications calls

table 6-7

file

See also Communications: Intersystem Communications Function Programmer's Guide, SC41-9590

commands for managing ICF file 5-1

description 5-1

object example E-1

waiting to use 5-8

writing application programs 1-1

interface

application considerations 7-1

language operations

chart A-1

operation

close 5-11

get-attributes 5-10

read 5-7

read-from-invited-program-devices 5-7

release 5-10

program

example of objects used E-1

file object example E-1, E-2, E-3

return code 0014

turnaround indication E-13

introduction

AS/400 APPC support 1-1

- invalid password attempts using APPC 3-12
- invite function 5-7
- ISDN (integrated services digital network)**
 - connection list 2-1
 - definition 1-3
 - network interface description 2-1

K

keyword

- chart A-2
- DDS security 3-13

L

language operation

- chart A-1

line description

- nonswitched connection example D-4
- switched connection example D-1
- X.21 short-hold mode example D-6, D-8

location parameter

- device description 3-4
- local location name 3-4
- mode 3-4
- remote location name 3-4
- remote network ID 3-4

location security

- description 3-8

low-entry networking node

- APPN network limitations 2-4
- definition 2-4

LU definition verb

- DEFINE_LOCAL_LU C-2
- DEFINE_MODE C-2
- DEFINE_REMOTE_LU C-2
- DISPLAY_LOCAL_LU C-3
- DISPLAY_MODE C-3
- DISPLAY_REMOTE_LU C-3

LU type 6.2 architecture

See also *Systems Application Architecture**
Common Programming Interface Communications
Reference, SC26-4399

- ACTIVATE_SESSION session control verb C-2
- ALLOCATE basic conversation verb C-12
- AS/400 control operator verbs C-1
- AS/400 system implementation C-1
- basic conversation verb
 - ALLOCATE C-12
 - CONFIRM C-13
 - CONFIRMED C-13
 - FLUSH C-14
 - GET_ATTRIBUTES C-14
 - POST_ON_RECEIPT C-16
 - PREPARE_TO_RECEIVE C-16
 - RECEIVE_AND_WAIT C-17
 - RECEIVE_IMMEDIATE C-18
 - REQUEST_TO_SEND C-18
 - SEND_DATA C-18
 - SEND_ERROR C-19

LU type 6.2 architecture (continued)

- basic conversation verb (continued)
 - TEST C-19
- change-number-of-sessions verbs C-1
- CHANGE_SESSION_LIMIT control operator verb C-1
- CONFIRM basic conversation verb C-13
- CONFIRMED basic conversation verb C-13
- control operator verb, AS/400
 - change-number-of-sessions verbs C-1
 - CHANGE_SESSION_LIMIT C-1
 - option sets C-26
 - verbs, change-number-of-sessions C-1
- conversation verb
 - basic C-12
 - option sets C-23
- CPI Communications 6-14
- DEACTIVATE_SESSION session control verb C-2
- DEFINE_LOCAL_LU definition verb C-2
- DEFINE_MODE definition verb C-2
- DEFINE_REMOTE_LU definition verb C-2
- definition verb C-2
- DISPLAY_LOCAL_LU definition verb C-3
- DISPLAY_MODE definition verb C-3
- DISPLAY_REMOTE_LU definition verb C-3
- FLUSH basic conversation verb C-14
- GET_ATTRIBUTES basic conversation verb C-14
- GET_TP_PROPERTIES verb C-19
- GET_TYPE verb C-20
- ICF implementation C-3
- INITIALIZE_SESSION_LIMIT control operator verb C-1
- LU definition verb
 - DEFINE_LOCAL_LU C-2
 - DEFINE_MODE C-2
 - DEFINE_REMOTE_LU C-2
 - DISPLAY_LOCAL_LU C-3
 - DISPLAY_MODE C-3
 - DISPLAY_REMOTE_LU C-3
- mapped conversation verb C-3
 - MC_ALLOCATE C-3
 - MC_CONFIRM C-4
 - MC_CONFIRMED C-5
 - MC_DEALLOCATE C-5
 - MC_FLUSH C-6
 - MC_GET_ATTRIBUTES C-6
 - MC_POST_ON_RECEIPT C-6
 - MC_PREPARE_TO_RECEIVE C-6
 - MC_RECEIVE_AND_WAIT C-6
 - MC_RECEIVE_IMMEDIATE C-10
 - MC_REQUEST_TO_SEND C-10
 - MC_SEND_DATA C-10
 - MC_SEND_ERROR C-11
 - MC_TEST C-11
- MC_ALLOCATE mapped conversation verb C-3
- MC_CONFIRM mapped conversation verb C-4
- MC_CONFIRMED mapped conversation verb C-5
- MC_DEALLOCATE mapped conversation verb C-5

LU type 6.2 architecture (continued)

MC_FLUSH mapped conversation verb C-6
MC_GET_ATTRIBUTES mapped conversation verb C-6
MC_POST_ON_RECEIPT mapped conversation verb C-6
MC_PREPARE_TO_RECEIVE mapped conversation verb C-6
MC_RECEIVE_AND_WAIT mapped conversation verb C-6
MC_RECEIVE_IMMEDIATE mapped conversation verb C-10
MC_REQUEST_TO_SEND mapped conversation verb C-10
MC_SEND_DATA mapped conversation verb C-10
MC_SEND_ERROR mapped conversation verb C-11
MC_TEST mapped conversation verb C-11
miscellaneous verb
 GET_TP_PROPERTIES C-19
 GET_TYPE C-20
 WAIT C-20
option sets for LU 6.2 control operator verb C-26
option sets for LU 6.2 conversation verb C-23
POST_ON_RECEIPT basic conversation verb C-16
PREPARE_TO_RECEIVE basic conversation verb C-16
PROCESS_SESSION_LIMIT control operator verb C-1
RECEIVE_AND_WAIT basic conversation verb C-17
RECEIVE_IMMEDIATE basic conversation verb C-18
relationship to APPC support 1-1
REQUEST_TO_SEND basic conversation verb C-18
RESET_SESSION_LIMIT control operator verb C-2
return code mapping 6-14
return codes, mapping LU 6.2 to ICF C-20
SEND_DATA C-18
SEND_ERROR basic conversation verb C-19
session control verbs
 ACTIVATE_SESSION C-2
 DEACTIVATE_SESSION C-2
specifying the resource parameter C-3
TEST basic conversation verb C-19
verb
 control operator C-26
 miscellaneous C-19
 session control C-2
WAIT verb C-20

M

manual

AS/400 system ix
communications G-1
communications architecture G-1
customer G-1
installation G-1

manual (continued)

PC Support G-1
planning G-1
programming G-1
SAA CPI Communications G-2
system operation G-1

mapped conversation

CPI Communications 3-2
ICF 3-2

mapped conversation verb

MC_CONFIRM C-4
MC_CONFIRMED C-5
MC_DEALLOCATE C-5
MC_FLUSH C-6
MC_GET_ATTRIBUTES C-6
MC_POST_ON_RECEIPT C-6
MC_PREPARE_TO_RECEIVE C-6
MC_RECEIVE_AND_WAIT C-6
MC_RECEIVE_IMMEDIATE C-10
MC_REQUEST_TO_SEND C-10
MC_SEND_DATA C-10
MC_SEND_ERROR C-11
MC_TEST C-11

mapping between LU 6.2 verbs

See Systems Application Architecture Common Programming Interface Communications Reference, SC26-4399*

mapping LU type 6.2 and ICF return codes

table C-20

MC_ALLOCATE mapped conversation verb C-3
MC_CONFIRM mapped conversation verb C-4
MC_CONFIRMED mapped conversation verb C-5
MC_DEALLOCATE mapped conversation verb C-5
MC_FLUSH mapped conversation verb C-6
MC_GET_ATTRIBUTES mapped conversation verb C-6
MC_POST_ON_RECEIPT mapped conversation verb C-6
MC_PREPARE_TO_RECEIVE mapped conversation verb C-6
MC_RECEIVE_AND_WAIT mapped conversation verb C-6
MC_RECEIVE_IMMEDIATE mapped conversation verb C-10
MC_REQUEST_TO_SEND mapped conversation verb C-10
MC_SEND_DATA mapped conversation verb C-10
MC_SEND_ERROR mapped conversation verb C-11
MC_TEST mapped conversation verb C-11
menu-driven communications configuration 1-3
message B-4
miscellaneous verb
 GET_TP_PROPERTIES C-19
 GET_TYPE C-20
 WAIT C-20

mode description

See also Communications: Operating System/400 Communications Configuration Reference, SC41-0001*

mode description (*continued*)

definition 2-3

mode status

command description 4-6

displaying 4-6

N

network

APPC (without APPN) two-system 2-4

network interface description

definition 2-1

node type 2.1 architecture

relationship to APPC support 1-1

nonsecure location

definition 3-8

nonswitched connection

configuration example D-3

null password implementations

table 3-7

O

officer authority, security 3-13

open operation 5-4

open or acquire operation

ICF general considerations 7-1

operation

acquire 5-4

close 5-11

open 5-4

release 5-10

option set

control operator verb

table C-24

conversation verb

table C-20

output

ICF general considerations 7-2

output parameter

description 6-4

overflow data (OVRFLWDTA) parameter 5-3

Override Intersystem Communications Function File (OVRICFF) command

description 5-1

Override Intersystem Communications Function Program Device Entry (OVRICFDEVE) command

description 5-2

parameters 5-2

OVRICFDEVE (Override Intersystem Communications Function Program Device Entry) command 5-2

parameters 5-2

OVRICFF (Override Intersystem Communications Function File) command

description 5-1

P

parameter

ENDMOD command 4-3

STRMOD command 4-2

VRYCFG (Vary Configuration) command 4-1

pass-through, SNA

definition 1-2

password attempts 3-12

PC Support/400

See also *PC Support/400: DOS Installation and Administration Guide*, SC41-0006

See also *PC Support/400: OS/2 Installation and Administration Guide*, SC41-0007

definition 1-2

physical security

description 3-7

PIP (program initialization parameters) data 5-4

POST_ON_RECEIPT basic conversation verb C-16

PREPARE_TO_RECEIVE basic conversation verb C-16

prestart job

CPI Communications 8-2

prestart job entry

See also *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590

See also *Programming: Work Management Guide*, SC41-8078

description 7-4

format 7-4

PROCESS_SESSION_LIMIT control operator verb C-1

profile

See also *Security Reference*, SC41-8083

duplicate 3-12

program

example

CPI Communications F-1

ICF E-1

remote

definition E-1

program call

definition 6-4

program device entry command

description of parameters 5-2

program security

CPI Communications 3-15, 3-16

ICF 3-13

SECURITY(NONE) 3-16

SECURITY(SAME) 3-16

program start request

errors B-1

ICF prestart jobs 7-4

protocol requirements, SNA 1-1

pseudonym

See also *Systems Application Architecture* Common Programming Interface Communications Reference*, SC26-4399

definition 6-5, 6-6

files for high-level languages 6-6, 6-7

pseudonym (*continued*)
used for writing applications 6-6

R

read operation 5-7
read-from-invited-program-devices operation 5-7
reason code
rejected program start requests
table B-1
receive-confirm (RCVCONFIRM) response
indicator 5-11
receive-control-data (RCVCTLDTA) response
indicator 5-12
receive-detach (RCVDETACH) response
indicator 5-12
receive-fail (RCVFAIL) response indicator 5-11
receive-turnaround (RCVTRNRND) response
indicator 5-12
RECEIVE_AND_WAIT basic conversation verb C-17
RECEIVE_IMMEDIATE basic conversation verb C-18
release operation 5-10, 7-4
remote location name
considerations for using 3-4
device selection 3-4
remote program
definition E-1
Remove Intersystem Communications Function
Program Device Entry (RMVICFDEVE) command
description 5-2
request-to-write (RQSWRT) function
description 5-9
sending APPC SIGNAL 5-9
REQUEST_TO_SEND basic conversation verb C-18
RESET_SESSION_LIMIT control operator verb C-2
resource name
X.21 short-hold mode line D-7
respond-to-confirm (RSPCONFIRM) function 5-9
response indicator
receive-confirm 5-11
receive-control-data 5-12
receive-detach 5-12
receive-fail 5-11
receive-turnaround 5-12
Retrieve Configuration Status (RTVCFGSTS) command
See Communications: Management Guide,
SC41-0024
return code
*See also Systems Application Architecture**
Common Programming Interface Communications
Reference, SC26-4399
description B-4
major code 00 B-5
major code 02 B-9
major code 03 B-14
major code 04 B-18
major code 08 B-19
major code 11 B-19
major code 34 B-20

return code (*continued*)
major code 80 B-23
major code 81 B-27
major code 82 B-30
major code 83 B-37
mapping between CPI Communications and LU
6.2 6-14
mapping LU type 6.2 and ICF C-20
using 5-13
RMVICFDEVE (Remove Intersystem Communications
Function Program Device Entry) command 5-2
RPG/400 programming language
example program
CPI Communications local program for inquiry
applications F-35
CPI Communications remote program for inquiry
applications F-47
ICF local program for inquiry applications E-36
ICF remote program for inquiry
applications E-44
user-written applications 1-1
RTVCFGSTS (Retrieve Configuration Status) command
See Communications: Management Guide,
SC41-0024
running
APPC 4-1

S

SDLC (synchronous data link control)
communications line used by APPC 1-3
secure location
definition 3-8
security
application program 3-13, 3-15
considerations 3-11
CPI Communications 3-15
DDS keyword 3-13
general considerations 3-6, 3-11
ICF 3-13
invalid password attempts 3-12
level 3-7
location 3-6, 3-8
nonsecure location 3-8
physical 3-6, 3-7
resource 3-6
session level 3-7
special authority (security officer and
service) 3-13
user ID 3-6
validation tables for establishing a session 3-8
SECURITY DDS keyword
specified with evoke function 5-5
security level
See also Security Reference, SC41-8083
in an APPC network 3-7
see ='
SEND_DATA basic conversation verb C-18

SEND_ERROR basic conversation verb C-19

sense data B-4
 See also *Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188

session
 description 3-1
 ending APPC
 using close operation 5-10
 using end-of-session function 5-10
 using release operation 5-10
 establishing APPC
 using open and acquire operations 5-4

session control verb
 ACTIVATE_SESSION C-2
 DEACTIVATE_SESSION C-2

session level security
 description 3-7

side information 6-1

SNA (Systems Network Architecture)
 See also *Communications: Distribution Services Network Guide*, SC41-9588
 definition 1-2
 distribution services 1-2
 protocol requirements 1-1

SNA Distribution Services (SNADS) 1-2
 See also *Communications: Distribution Services Network Guide*, SC41-9588

SNA FMH7 sense data
 table B-1

SNA pass-through
 See also *Communications: Remote Work Station Guide*, SC41-0002
 definition 1-2

SNADS (SNA Distribution Services) 1-2
 See also *Communications: Distribution Services Network Guide*, SC41-9588

special authority
 security 3-13

Start Mode (STRMOD) command
 parameters 4-2

starting
 communications support 4-1
 session
 open and acquire operation 5-4
 transaction
 evoke function 5-4

STRMOD (Start Mode) command
 parameters 4-2

subsystem description
 prestart job entry 7-4

support provided by APPC
 communications lines 1-3
 network management 1-3

switched connection
 configuration example D-1

synchronization level
 specified with evoke function 5-5

synchronous conversation
 definition 3-1

synchronous data link control (SDLC)
 communications line used by APPC 1-3

SYNLVL keyword
 See synchronization level

syntax, command
 See *Programming: Control Language Reference*, SC41-0030

system name
 important information 2-3

system-supplied format
 See also *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590
 \$\$EOS system-supplied format A-3
 \$\$EVOK system-supplied format A-3
 \$\$EVOKET system-supplied format A-3
 \$\$EVOKNI system-supplied format A-3
 \$\$FAIL system-supplied format A-3
 \$\$RCD system-supplied format A-3
 \$\$SEND system-supplied format A-3
 \$\$SENDET system-supplied format A-3
 \$\$SENDNI system-supplied format A-3
 \$\$TIMER system-supplied format A-3
 security 3-15

Systems Network Architecture (SNA)
 See also *Communications: Distribution Services Network Guide*, SC41-9588
 definition 1-2
 distribution services 1-2
 protocol requirements 1-1

T

target system
 summary of user IDs 3-11

TEST basic conversation verb C-19

timer function 5-10

token-ring network
 communications line used by APPC 1-3

Trace ICF (TRCICF) function
 description 7-4, 8-2

transaction
 APPC conversation 3-2
 definition 3-2, 5-4
 starting 5-4

turnaround indication
 ICF return code 0014 E-13

U

underscore (_)
 used in phrases 6-5

unit-of-work identifier
 definition 3-6

user ID
 target system 3-11

user-written application

- using CPI Communications calls 1-2
- using ICF files 1-1

V**validation table 3-8****variable buffer management (VARBUFMGMT)****function 5-6, 5-7****Vary Configuration (VRYCFG) command**

See also Communications: Management Guide,
SC41-0024

example 4-2

parameters 4-1

vary off 4-1**vary on 4-1****verb**

- ALLOCATE basic conversation C-12
- basic conversation C-12
- change-number-of-sessions
 - control operator C-1
 - types of C-1
- change-number-of-sessions C-1
- control operator
 - option sets C-26
- conversation
 - option sets C-24
- DEACTIVATE_SESSION session control C-2
- DEFINE_LOCAL_LU definition C-2
- DEFINE_MODE definition C-2
- DEFINE_REMOTE_LU definition C-2
- DISPLAY_LOCAL_LU definition C-3
- DISPLAY_MODE definition C-3
- DISPLAY_REMOTE_LU definition C-3
- FLUSH basic conversation C-14
- GET_ATTRIBUTES basic conversation C-14
- GET_TP_PROPERTIES C-19
- GET_TYPE C-20
- INITIALIZE_SESSION_LIMIT control operator C-1
- LU definition C-2
- mapped conversation C-3
- MC_CONFIRM mapped conversation C-4
- MC_CONFIRMED mapped conversation C-5
- MC_DEALLOCATE mapped conversation C-5
- MC_FLUSH mapped conversation C-6
- MC_GET_ATTRIBUTES mapped conversation C-6
- MC_POST_ON_RECEIPT mapped conversation C-6
- MC_PREPARE_TO_RECEIVE mapped
 - conversation C-6
- MC_RECEIVE_AND_WAIT mapped
 - conversation C-6
- MC_RECEIVE_IMMEDIATE mapped
 - conversation C-10
- MC_SEND_DATA mapped conversation C-10
- MC_SEND_ERROR mapped conversation C-11
- MC_TEST mapped conversation C-11
- miscellaneous C-19
- POST_ON_RECEIPT basic conversation C-16
- PREPARE_TO_RECEIVE basic conversation C-16

verb (continued)

- PROCESS_SESSION_LIMIT control operator C-1
- RECEIVE_AND_WAIT basic conversation C-17
- RECEIVE_IMMEDIATE basic conversation C-18
- REQUEST_TO_SEND basic conversation C-18
- RESET_SESSION_LIMIT control operator C-2
- SEND_DATA basic conversation C-18
- SEND_ERROR basic conversation C-19
- session control C-2
- WAIT C-20

VRYCFG (Vary Configuration) command 4-1

example 4-2

W**WAIT verb C-20****Work with Configuration Status (WRKCFGSTS) command**

See Communications: Management Guide,
SC41-0024

write operation 5-6**writing**

APPC application program 5-1

CPI Communications 6-1

ICF 5-1

application program

See also Communications: Intersystem Communications Function Programmer's Guide,
SC41-9590

See also Systems Application Architecture Common Programming Interface Communications Reference,* SC26-4399

using CPI Communications calls 1-2

using ICF files 1-1

CPI application programs

See Systems Application Architecture Common Programming Interface Communications Reference,* SC26-4399

WRKCFGSTS (Work with Configuration Status) command

See Communications: Management Guide,
SC41-0024

X**X.21 short-hold mode**

configuration example D-6

resource names D-7

X.25

packet-switching data network

communications line used by APPC 1-3



Program Number: 5738-SS1

Printed in Denmark by
Scanprint a/s, Viby J.

SC41-8189-01

