AS/400 Advanced Series

# APPC Programming

*Version 4*

SC41-5443-00

AS/400 Advanced Series

# APPC Programming

*Version 4*

SC41-5443-00

**Note**

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

**First Edition (August 1997)**

This edition applies to the licensed program IBM Operating System/400 (Program 5769-SS1), Version 4 Release 1 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions.

Make sure that 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. If you live in the United States, Puerto Rico, or Guam, you can order publications through the IBM Software Manufacturing Solutions at 1-800-879-2755. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication. You can also mail your comments to the following address:

IBM Corporation
Attention Department 542
IDCLERK
3605 Highway 52 N
Rochester, MN 55901-7829 USA

or you can fax your comments to:

United States and Canada: 1-800-937-3430
Other countries: 1-507-253-5192

If you have access to the Internet, you can send your comments electronically to IDCLERK@RCHVMW2.VNET.IBM.COM; IBMMAIL, to IBMMAIL(USIB56RZ).

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

# Contents

# 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 that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact the software interoperability coordinator. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee. Address your questions to:

IBM Corporation
Software Interoperability Coordinator
3605 Highway 52 N
Rochester, MN 55901-7829
U.S.A.

This publication could contain technical inaccuracies or typographical errors.

This publication may refer to products that are not currently available. 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.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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.**

## Trademarks

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

ACF/VTAM
Advanced Peer-to-Peer Networking
AIX
AnyNet
AnyNet/2
AnyNet/400
Application System/400
APPN
AS/400
C/400
IBM
ILE
NetView
Novell
Opticonnect for OS/400
Operating System/400
OS/400
PS/2
VTAM
400

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

# About APPC Programming (SC41-5443-00)

This book 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 book are application program considerations, configuration requirements and commands, problem management for APPC, and general networking considerations.

Although this book 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.

## Who Should Read This Book

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

You should have knowledge of general communications concepts. For information on basic communications, you can 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:

- *System Operation*
- *ICF Programming*
- *CPI Communications Reference*

## Prerequisite and Related Information

For information about Advanced 36 publications, see the *Advanced 36 Information Directory*, SC21-8292, in the AS/400 Softcopy Library.

For information about other AS/400 publications (except Advanced 36), see either of the following:

- The *Publications Reference*, SC41-5003, in the AS/400 Softcopy Library.
- The AS/400 Information Directory, a unique, multimedia interface to a searchable database that contains descriptions of titles available from IBM or from selected other publishers.

For a list of related publications, see the “Bibliography” on page X-1.

## Information Available on the World Wide Web

More AS/400 information is available on the World Wide Web. You can access this information from the AS/400 home page, which is at the following universal resource locator (url) address:

`http://www.as400.ibm.com`

Select the Information Desk, and you will be able to access a variety of AS/400 information topics from that page.

# 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. APPC application programs can also communicate over lines using the Internet Protocol (IP) of Transmission Control Protocol/Internet Protocol (TCP/IP).

## 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 the following program interfaces:

- The intersystem communications function (ICF) file interface
- The Common Programming Interface (CPI) Communications call interface
- The CICS file interface
- The sockets application program interface (API)

In ICF, the LU 6.2 verbs are implemented using data description specifications (DDS) keywords and system-supplied formats. 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.

In CPI Communications, the LU 6.2 verbs are implemented using CPI Communications calls. Refer to “Mapping of CPI Communications Calls to ICF Operations and Functions” on page 6-6 for a mapping of CPI Communications calls and ICF operations and functions. The *CPI Communications Reference* book provides a mapping between LU 6.2 verbs and CPI Communications calls.

In CICS/400 support, the LU 6.2 verbs are implemented using EXEC CICS commands. For a list of the EXEC CICS commands used to control an APPC conversation, see the *CICS/400 Administration and Operations Guide*.

For the sockets API, the LU 6.2 verbs are implemented using the socket functions. For a list of the sockets APIs, see the *Sockets Programming* book.

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 Support* book provides details about the APPN support.

## Relationship between APPC and AnyNet/400 Support

AnyNet is a family of products that allow applications written for one type of network protocol to be run over a different type of network protocol.

Without AnyNet, your choice of application program interface (API) dictates your choice of network protocol, or the opposite, your choice of network protocol dictates your choice of APIs. For example, without AnyNet, an ICF or CPI Communications (APPC) application program can only run over an SNA network. To run both ICF or CPI Communications (APPC) and sockets programs without AnyNet, requires both an SNA and a TCP/IP network.

AnyNet gives you more options. AnyNet allows you to mix and match your choice of application programs with your choice of network protocols. In fact, you can do this without changing your application programs. Your choice of destination addresses (such as remote location) determines which type of network protocol is used.

AnyNet/400 is included with the Operating System/400 licensed program and provides the following support:

- APPC over TCP/IP

  The APPC over TCP/IP support allows APPC application programs (such as ICF or CPI Communications applications) to communicate between systems in a TCP/IP network. The systems running the APPC application programs must both have APPC over TCP/IP support. For

more information on APPC over TCP/IP support, see the *Communications Configuration* book.

- Sockets over SNA

  The sockets over SNA support allows sockets application programs to communicate between systems in an SNA network. The systems running the sockets application programs must both have sockets over SNA support unless there is an AnyNet gateway between them. With an AnyNet gateway (such as AnyNet/2 Sockets over SNA Gateway) between an SNA and a TCP/IP network, sockets application programs running on systems with sockets over SNA support can communicate with sockets application programs running on systems in the TCP/IP network.

IBM-supplied application programs that use the CPI Communications, ICF, or sockets interfaces (such as display station pass-through or File Transfer Protocol (FTP)) can also take advantage of AnyNet support. Therefore, you do not need to have user-written applications to take advantage of AnyNet 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
- Use the EXEC CICS commands
- Use the sockets APIs

If you use ICF, you can write application programs using the following languages.

- Integrated Language Environment C/400 (ILE C/400) language
- Integrated Language Environment COBOL/400 (ILE COBOL/400) language
- FORTRAN/400 language
- Integrated Language Environment RPG/400 (ILE RPG/400) language

Chapter 5 of this guide and the *ICF Programming* book contain information about writing applications using ICF files.

If you use CPI Communications, you can write application programs in the following programming languages:

- ILE C/400 language
- ILE COBOL/400 language
- FORTRAN/400 language
- REXX/400
- ILE RPG/400 language
- Cross System Product (CSP)

Chapter 6 of this guide and the *CPI Communications Reference* book contain information about writing applications using the CPI Communications call interface.

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

If you use sockets, you can write application programs using the ILE C/400 language. Both the source and target programs must use the sockets API. For more information about writing sockets application programs, see the *Sockets Programming* book.

**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, CICS/400, and sockets interfaces. However, when communicating with another program, an APPC application cannot use combinations of ICF operations or functions and CPI Communications calls, EXEC CICS commands, or sockets APIs 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 *Distributed Data Management* book for additional information.
- **Distributed Relational Database Architecture (DRDA) support**, provides access to relational data across systems that support this architecture. The operating system and the DB2/400 Query Manager and SQL Development Kit licensed program combine to provide this support on AS/400 systems. Refer to the *Distributed Database Programming* book 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 *SNA Distribution Services* book 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 Support* book 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 Programming* book for additional information.
- **Client Access/400**, which provides system functions to an attached personal computer. Refer to "Client Access/400 Books" on page X-1 for a list of Client Access/400 books.
- **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 *DSNX Support* book 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 *Local Device Configuration* book for additional information.
- **CICS/400 support**, which allows user-written application programs to process transactions entered at remote work stations. CICS/400 support 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.
- **APING support**, a function of the operating system provided by the Verify APPC Connection (VFYAPPCCNN) command, also known as APING, exchanges data packets between the local location and the specified remote location using Advanced Program-to-Program Communications (APPC), and measures the round-trip time of each data exchange.

  For this function to work, the remote location specified must be running the target portion of this function, **APINGD** (APING daemon). Refer to the *CL Reference* book for additional information.
- **AREXEC support**, a function of the operating system provided by the Run Remote Command (RUNRMTCMD) command, also known as AREXEC when an SNA address is specified for the remote location name, allows AS/400 users to run a command on a remote system that is running the target portion of this function.

  The target portion of this function can be an *rexecd* (remote executing) daemon if you specify *IP for the address type, or an *AREXECD* (APPC remote executing) daemon when an SNA address is specified for the remote location name. If you want to run a command on a Client Access/400 client that supports this function, you will need to specify an SNA address for the remote location name.

  When the command is sent to the remote system, the local system waits for the command to complete and the output from the remote command will be sent back to the local system and placed in a spooled file. Refer to the *CL Reference* book for additional information.
- **OptiConnect support for APPC**

  OptiConnect optical fiber connectivity among adjacent machines, supports:

  – Two-phase commitment control
  – Loosely coupled parallelism
  – All APPC applications that use the ICF or CPI/C interfaces with the value APPN = *NO.

Using the optical bus leads to enhanced performance. You can use this support to run many types of APPC applications across the optical bus. This will decrease complexity and increase performance.

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 *Communications Configuration* book contain additional information concerning configurations.

The following TCP/IP applications are some of the TCP/IP applications that use the sockets interface:

- File Transfer Protocol (FTP)
- line printer daemon (LPD)
- line printer requester (LPR)
- Simple Mail Transfer Protocol (SMTP)
- Simple Network Management Protocol (SNMP)
- Remote Procedure Call (RPC)

TELNET and the PASCAL APIs do not use the sockets interface. TELNET and applications that use the PASCAL APIs cannot use APPC support.

## 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 *Communications Configuration* book, which also has a description of the communications configuration process.

## 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):

- Distributed data interface (DDI)
- Frame-relay network
- FSIOP
- IDLC (ISDN data link control) [1] switched or non-switched
- SNA pass-through
- 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
- Wireless network

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

- If an APPC controller 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 controller 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
  - **Distributed host command facility (DHCF)** [6] support
  - **Network routing facility (NRF)** [7] support
  - **SNA pass-through**
  - **SNA Primary LU2 Support (SPLS)** [8]

  They can all be active on the line at the same time.
- If an APPC controller 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
  - SNA pass-through
  - SNUF
  - DSNX support
  - DHCF support
  - NRF support
  - SPLS
  - **Transmission Control and Protocol/Internet Protocol (TCP/IP)**.[9]
- If an APPC controller is configured to use a token-ring network, then multiple remote systems can be active at

---

1 ISDN is the abbreviation 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 NRF is a function of the operating system that supports the data link between a System/370 terminal using an AS/400 application in an NRF (Network Routing Facility) environment.

8 SPLS is a function of the operating system that supports the data link between a System/370 terminal using an AS/400 application in an SNA environment. The HCF and NRF licensed programs are not required for SPLS.

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

the same time. The APPC support can share the token-ring network line with:

  - Finance support
  - Remote work station support
  - The SNA version of RJE
  - 3270 device emulation for SNA
  - SNUF
  - DSNX support
  - DHCF support
  - NRF support
  - SNA pass-through
  - SPLS
  - TCP/IP

- If an APPC controller is configured to use Ethernet support, then multiple remote systems can be active at the same time. The APPC support can share the Ethernet line with:

  - Finance support
  - Remote work station support
  - 3270 device emulation for SNA
  - SNA pass-through
  - SNUF
  - DSNX support
  - DHCF support
  - NRF support
  - SPLS
  - TCP/IP

- If an APPC controller is configured to use DDI support, then multiple remote systems can be active at the same time. The APPC support can share the DDI line with:

  - Remote work station support
  - DHCF support
  - NRF support
  - SNA pass-through
  - SPLS
  - TCP/IP

- If an APPC controller is configured to use a frame-relay network, then multiple remote systems can be active at the same time. The APPC support can share the frame-relay network line with:

  - DHCF support
  - NRF support
  - SNA pass-through
  - SPLS
  - TCP/IP

- If an APPC controller is configured to use a wireless network, then multiple remote systems can be active at the same time. The APPC support can share the wireless network line with:

  - SNA pass-through
  - TCP/IP

- If an APPC controller is configured to use IDLC, the APPC support can share the IDLC line with:

  - Remote work station support
  - 3270 device emulation for SNA
  - SNA pass-through
  - SNUF
  - DSNX

# 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 Support* book, the *Communications Configuration* book, 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.

For frame-relay, you must also create a network interface description.

To configure devices, controllers, and lines you must have *IOSYSCFG special authority in your user profile for the create and change configuration commands.

Without requiring this special authority it is possible for anyone on the system to set up an alternate configuration that they can use to bypass the security on the system. Refer to the *System Operation*book for more information.

## Configuring for Sockets

To have TCP/IP sockets applications communicate across an SNA connection, requires the following additional configuration steps. These steps can be done using the Configure AF_INET Sockets over SNA (CFGIPS) command.

1. Designate the local IP address or addresses assigned to the local host using the Add IP over SNA Interface (ADDIPSIFC) command.
2. Designate the IP routes to the remote hosts local host using the Add IP over SNA Route (ADDIPSRTE) command.
3. Designate the SNA location names that are associated with each IP address for each remote host using the Add IP over SNA Location Entry (ADDIPSLOC) command.

The *Sockets Programming* book contains information about configuring for sockets.

## Configuring for APPC Over TCP/IP Support

Using APPC over TCP/IP support requires that you set the allow ANYNET support (ALWANYNET) network attribute to *YES. To run APPC over a TCP/IP network, you must configure the APPC controller, device, and mode descriptions as always. However, the line description is not associated with these configuration objects. The line description must be associated with a network controller. The association of APPC locations and TCP/IP addresses must also be performed. The *TCP/IP Configuration and Reference* book contains more information about configuring a TCP/IP network. The *Communications Configuration* book contains a configuration example for APPC over TCP/IP support.

## 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 Support* book 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 (CRTCNNL)
   - Change Connection List (CHGCNNL)
   - Display Connection List (DSPCNNL)
   - Delete Connection List (DLTCNNL)
   - Work with Connection Lists (WRKCNNL)
   - Add Connection List Entry (ADDCNNLE)
   - Change Connection List Entry (CHGCNNLE)
   - Rename Connection List Entry (RNMCNNLE)
   - Remove Connection List Entry (RMVCNNLE)

- 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)

## Configuring for Frame-Relay

Using frame-relay requires that you configure a network interface description before the line, controller, device, and mode descriptions. The *LAN and Frame Relay Support* book contains more information about configuring for frame-relay.

The network interface description describes the interface between the AS/400 system and the frame-relay network. The following commands are used when working with network interface descriptions:

- Create Network Interface Description (Frame-Relay Network) (CRTNWIFR)
- Change Network Interface Description (Frame-Relay Network) (CHGNWIFR)
- 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 (Distributed Data Interface) (CRTLINDDI)
- Change Line Description (Distributed Data Interface) (CHGLINDDI)
- Create Line Description (Frame-Relay Network) (CRTLINFR)
- Change Line Description (Frame-Relay Network) (CHGLINFR)
- Create Line Description (IDLC) (CRTLINIDLC)
- Change Line Description (IDLC) (CHGLINIDLC)
- Create Line Description (SDLC) (CRTLINSDLC)
- Change Line Description (SDLC) (CHGLINSDLC)
- 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)
- Create Line Description (Wireless) (CRTLINWLS)
- Change Line Description (Wireless) (CHGLINWLS)
- 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 IP Networks:** If you are configuring for APPC over IP, use the value *ANYNW for the link type (LINKTYPE) parameter for these commands.

**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 “Programs Communicating on the Same System—Configuration Example” on page D-6.

## 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* book 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 “Programs Communicating on the Same System—Configuration Example” on page D-6.

## 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.

**Note:** 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

The following 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).

**QCASERVR** a mode for use with AS/400 server functions.

**QRMTWSC** a mode tailored for use with the 5494 remote work station controller.

**QSPWTR** a mode used for Advanced Function Printing (AFP) support.

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 support and from the APPN remote location list for configurations using APPN support. Refer to the *Communications Configuration* book 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 *Communications Configuration* book 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 (DLTDEVD)
- Delete Mode Description (DLTMODD)
- Delete Connection List (DLTCNNL)

## Configuration for an APPC Network without APPN Support

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 Support* book.

In this kind of network, you must configure an APPC device description for each end point.

**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 start (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 choose a particular session over which 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 end 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 **communi-**

**cations 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* book 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 (VARBUFMGT) 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 Table 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.

*Table 3-1. General Data Stream Structure*

| Byte | Name | Description |
|---|---|---|
| 0,1 | LL | 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.<br><br>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. |
| 2,3 | GDS ID | 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.<br><br>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. |
| 4 - n | User data | Unarchitected user data. |

[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.

***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 VARBUFMGT function is *not* used. For information about basic conversations in which the VARBUFMGT function *is* used, refer to "Variable-Buffer-Management Function" on page 5-6.

The AS/400 system treats the LL as user data. If VARBUFMGT 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*.

**Conversation States:** The LU 6.2 verbs that a program can issue for a particular conversation depend on the state of the conversation. As the program issues verbs, the state of the conversation can change. The state of the conversation can change based on the following:

- The function of the verb.
- The result of a verb issued by the remote program.
- The result of network errors.

A program can use the ICF get-attributes operation to get its conversation state. Table C-1 on page C-11 shows the attributes returned by the get-attributes operation including the possible values for the conversation state. The *SNA Transaction Programmer's Reference Manual for LU Type 6.2* has detailed information about conversation states.

## 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 except when using advanced peer-to-peer networking (APPN) support, or when running over a TCP/IP network. In general, APPN support 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 support, it is done through a configuration list. In either case, the system creates device descriptions containing the remote location name when APPN support 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 non-switched 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, 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, 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 **protected logical unit of work identifier** (LUWID) is the unit-of-work identifier used to keep track of distributed two-phase commit resources. A two-phase commit transaction can have multiple commits and rollbacks issued during the transaction. A new protected logical unit of work is started after each successful commit and rollback operation. Thus, multiple LUWIDs can be assigned for one transaction.

The unit-of-work identifier and protected LUWID can be found in four places within the system. When the APPC job is running, the unit-of-work identifier and protected LUWID can be displayed by entering the Display Job (DSPJOB) command for that job. The protected LUWID can also be displayed, when the APPC job is running, by entering the Work with Commitment Definitions (WRKCMTDFN) command. When an APPC job ends, the unit-of-work identifier or protected LUWID is printed on the AS/400 job log. To use user-defined auditing, a message (CPI835D or CPI9803) is sent to the history file. These messages contain the job name and the unit-of-work identifier or protected LUWID. Even after the jobs have ended, the unit-of-work identifier (or protected LUWID) 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 or protected LUWID 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. The AS/400 system displays the protected LUWID when option 16 is selected from the Display Job display. It also displays the protected LUWID when option 5 is selected from the Work with Commitment Definitions display. For distributed transactions that involve an AS/400 system to another AS/400 system, the unit-of-work identifier or protected LUWID 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. Or select option 16 from the Display Job display to display the protected LUWID. The unit-of-work identifier or protected LUWID shown at both systems is 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. You can use the WRKCMTDFN command and specify the protected LUWID from the job on the remote system to find the job on the local system.

[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.

## Two-Phase Commit

Using **two-phase commit** (known in SNA as sync point) protocols, customer applications can perform transactions that involve updates to data stored on two or more systems. Within a transaction, all of the updates for that transaction must be committed or rolled back together.

An application can invoke two-phase commit protocols to define synchronization points. At each synchronization point, all protected resources (such as files or databases), which are used on multiple systems, are in consistent states, even if errors or failures have occurred. All updates to the protected resources must be either committed or rolled back to the last synchronization point.

Two-phase commit also defines a protocol to recover from system, conversation, transaction program (TP), and local resource failures. In other words, two-phase commit provides the ability to resynchronize resources after such a failure.

## Protected Conversations and Resources

In order to use the two-phase commit protocols, the application must start commitment control and evoke a protected conversation. ICF application programs need to evoke a conversation at a synchronization level of commit (SYNLVL(*COMMIT)) for two-phase commit. CPI Communications application programs need to evoke a conversation at a synchronization level of sync point (CM_SYNC_POINT).

Conversations that use two-phase commit protocols are also known as **protected conversations**. Protected conversations are not allowed in the System/36 and System/38 environments of an AS/400 system. They are also not allowed for single-session connections.

A network of application programs communicating over protected conversations forms a **transaction program network** as shown in Figure 3-1. The **initiator** issues the first commit operation. Once the commit is issued, each node in the transaction program network is asked to commit its resources using the two-phase commit protocols. If all agents respond with a commit operation, all nodes in the transaction program network commit their updates. If one node responds with a rollback, all of the nodes in the transaction program network are forced to roll back their changes to the last synchronization point.

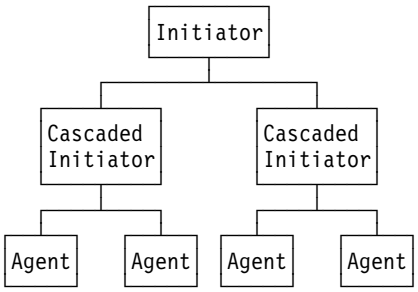



*Figure 3-1. Transaction Program Network*

The **two-phase commit** protocol permits updates to protected resources to be committed or rolled back as a unit. During the first phase, agents are asked if they are ready to commit. If all agents respond positively, they are asked to commit their updates. Otherwise, the agents are asked to roll back their updates.

## Resynchronization

Resynchronization of protected resources happens when a failure (such as a session or node failure) occurs during a commit or rollback operation. Normal communications recovery methods should be used to correct the communications error. After the communications error is corrected, the system brings up a session between the two systems that detected the error. Then, it resynchronizes the protected resource states.

For more information about two-phase commit and resynchronization, see the *Backup and Recovery* book.

## APPC Data Compression

APPC session-level data compression reduces the amount of data sent across a communications line during an APPC session. It can increase the throughput on slower lines. It can reduce the cost per bit on expensive lines. However, data compression also uses processing unit cycles. It can actually reduce throughput on fast lines that can send the data faster than the processing unit can compress it. Data compression varies in its effectiveness depending on the content of the data. For example, data compression is more effective on text than on binary data.

You can use APPC data compression between any two systems that support APPC and data compression. For example, clients using Client Access/400 can use APPC data compression if Communications Manager/2 or Communications Manager/400 and the AS/400 mode description are configured properly.

You can set up APPC to do compression in various ways. APPC can compress the outbound data, the inbound data, or both. You can select from two different compression algorithms and three levels of one of the compression algorithms.

In other words, you can select from four levels of data compression.

You can specify network attributes to set the system strategy for APPC data compression. For APPC sessions, the system can do the following:

- Require data compression
- Request data compression based on line speed
- Request data compression regardless of line speed
- Allow data compression
- Disallow data compression

This network attribute may be overridden by its corresponding parameter in a mode description. The system can also notify session end nodes that data compression should be done. This applies when the system is an APPN network node serving as an intermediate node for the session. This request may be based on line speed. This network attribute has no corresponding parameter in mode descriptions.

The compression algorithms are:

**Run-length encoding (RLE)** Substitutes a 1- or 2-byte sequence in the data stream for each repeated run of the same character. This algorithm requires no storage and fewer processing unit cycles than the adaptive dictionary-based compression algorithm. Its compression ratio is worse than the compression ratio for adaptive dictionary-based compression.

**Adaptive dictionary-based compression** Adaptive dictionary-based compression is a dynamic compression algorithm, similar to Lempel-Ziv, that compresses previously seen strings to 9-, 10-, and 12-bit codes. This algorithm is referred to as LZ. The algorithm is adaptive because it makes its codes by referring to previous instances of strings in the data stream. These coded strings are stored in a common dictionary that is created adaptively as the data flows from the sender to the receiver. The LZ algorithm requires storage and extra processing unit cycles.

Three levels of LZ compression are supported:

**LZ9** Requires less storage and fewer processing unit cycles than LZ10 and LZ12. However, its compression ratio is also worse than LZ10 and LZ12.

**LZ10** Requires more storage and processing unit cycles than LZ9 but less storage and fewer processing unit cycles than LZ12. Similarly, its compression ratio is better than LZ9 but worse than LZ12.

**LZ12** Requires the most storage and processing unit cycles and has the best compression ratio.

## Considerations for Data Compression

When deciding how to configure your system for APPC session-level data compression, you should consider the following:

**Line speed** The slower the line, the more likely the line is a performance bottleneck. Also, the slower the line, the likelier the data compression will improve the throughput of the line. The faster the line, the likelier the data is transmitted faster than it is compressed and decompressed.

**Processing unit utilization** Data compression and decompression use processing unit cycles at both ends of the session. If processing unit cycles are available at both ends of the session, data compression is viable. If the processing unit is heavily used, data compression may adversely affect system performance.

**Line charges** Data compression reduces the amount of data transmitted. Therefore, it should reduce utilization-based line charges (such as X.25).

**Line usage** Heavily-used lines can benefit from data compression because data compression can reduce the amount of traffic on the lines.

**Intermediate node compression requests** Intermediate nodes should request compression if sessions routed through them go in or out on a slow line. If System A is on a fast line but has sessions routed through intermediate nodes, System A should allow compression. By allowing compression, System A can compress data when an intermediate system requests it.

**Specialized modes** Unique compression needs should be specified using a unique mode description. Thus, the network attributes and commonly used modes can meet most needs, and a few specialized modes can meet any unique needs.

**Type of data** The effectiveness of the compression algorithms depends on the content of the data being compressed. RLE compresses repeated characters. Its effectiveness is entirely dependent on the quantity of repeated characters. For example, a compiled program may have only a few repeated characters. Similarly, the LZ compression algorithm compresses repeated strings of characters. Therefore, its effectiveness is dependent on the quantity of repeated character strings.

## Specifying Data Compression Parameters for a Mode Description

There are three mode description parameters that control APPC data compression. The first is the data compression (DTACPR) parameter. The DTACPR parameter specifies whether or not you want data compressed for sessions that have one end at this AS/400 system. The second is the inbound data compression (INDTACPR) parameter, which specifies the desired level of compression for inbound data. The third is the outbound data compression (OUTDTACPR) parameter, which specifies the desired level of compression for outbound data. These mode description parameters allow you to specify data compression settings, which can be unique to the session or sessions using this mode. These parameters are used with the Create Mode Description (CRTMODD) and the Change Mode Description (CHGMODD) commands.

The **DTACPR** parameter has an initial value of *NETATR, which means the parameter uses the value of the DTACPR network attribute. To disallow data compression on APPC sessions using this mode, specify *NONE. To allow data compression when session partners request it, specify *ALLOW. To request data compression for APPC sessions using this mode, specify *REQUEST. If you specify *REQUEST, data is compressed only if the session partner grants the request. To require data compression on APPC sessions, specify *REQUIRE. If you specify *REQUIRE, sessions using this mode are established only if the partner grants the request for compression at the specified compression levels. No other compression levels can be used for both inbound and outbound data. To request data compression based on line speed, specify the maximum line speed at which you want data compressed.

The **INDTACPR** and **OUTDTACPR** parameters have an initial value of *RLE. This means that the run-length encoding algorithm is used to compress inbound and outbound data. To get better compression ratios while using more storage and expending more processing unit cycles, specify *LZ9, *LZ10, or *LZ12. The LZ algorithm with the 9-bit code (*LZ9) has the worst compression ratio and the lowest resource usage of the LZ choices. On the other hand, *LZ12 has the best compression ratio and the highest resource usage.

## Specifying Network Attributes for Data Compression

There are two network attributes that control APPC data compression. One is the data compression (DTACPR) network attribute. The DTACPR network attribute specifies whether or not you want data compressed for sessions that have one end at this AS/400 system. The DTACPR network attribute is only used when the DTACPR parameter in a mode description is *NETATR. The second network attribute that affects APPC data compression is the intermediate node data compression (DTACPRINM) network attribute. The DTACPRINM network attribute specifies the level of data compression to request when this AS/400 system is an intermediate session node. These network attributes can be specified using the Change Network Attributes (CHGNETA) command.

The **DTACPR** network attribute has an initial value of *NONE. It can also have values of *ALLOW, *REQUEST, *REQUIRE, or a specific line speed. For more information on the meaning of these values, see the discussion of the DTACPR parameter in "Specifying Data Compression Parameters for a Mode Description."

The **DTACPRINM** network attribute has an initial value of *NONE, which means that when this AS/400 system is an APPN intermediate node, this system does not notify the end nodes that data compression is needed. To notify the end nodes that data compression should be done, specify *REQUEST. To base requests for data compression on line speed, specify the maximum line speed at which you want data compressed. If data compression is done, the end nodes (not this intermediate node) do the compression and decompression. For data to be compressed, the end nodes must have their modes configured to allow it.

**How to Get the Compression You Want:** Table 3-2 shows whether or not data is compressed based on the values of the data compression (DTACPR) parameters at the two end points.

*Table 3-2 (Page 1 of 2). Results of Compression Settings. This table shows whether or not data is compressed depending on the data compression (DTACPR) parameters of System A and System B.*

| | DTACPR Value | System B | | | |
|---|---|---|---|---|---|
| | | *NONE | *ALLOW | *REQUEST | *REQUIRE |
| System A | *NONE | 1 | 1 | 1 | 5 |
| | *ALLOW | 1 | 2 | 3 | 7 |
| | *REQUEST | 1 | 3 | 3 | 7 |
| | *REQUIRE | 4 | 6 | 6 | 8 |

*Table 3-2 (Page 2 of 2). Results of Compression Settings. This table shows whether or not data is compressed depending on the data compression (DTACPR) parameters of System A and System B.*

| | DTACPR Value | System B | | | |
|---|---|---|---|---|---|
| | | ***NONE** | ***ALLOW** | ***REQUEST** | ***REQUIRE** |

**Notes:**

1. No compression for either inbound or outbound data.
2. No compression for either inbound or outbound data unless an intermediate node requests compression. See note 3 for information about the compression levels used for the session.
3. The compression level used from System A to System B is the minimum of the outbound level of System A (OUTDTACPR) and the inbound level of System B (INDTACPR). The compression level used from System B to System A is the minimum of the outbound level of System B (OUTDTACPR) and the inbound level of System A (INDTACPR).
4. The session is not established unless the inbound (INDTACPR) and outbound (OUTDTACPR) compression levels of System A are both *NONE. In which case the session is established, but the data is not compressed.
5. The session is not established unless the inbound (INDTACPR) and outbound (OUTDTACPR) compression levels of System B are both *NONE. In which case the session is established, but the data is not compressed.
6. The inbound data compression level (INDTACPR) for System B must be greater than or equal to the outbound data compression level (OUTDTACPR) for System A. Likewise, the outbound data compression level (OUTDTACPR) for System B must be greater than or equal to the inbound data compression level (INDTACPR) for System A. If these two conditions are met, the compression levels for System A are used. Otherwise, no session is established.
7. The inbound data compression level (INDTACPR) for System A must be greater than or equal to the outbound data compression level (OUTDTACPR) for System B. Likewise, the outbound data compression level (OUTDTACPR) for System A must be greater than or equal to the inbound data compression level (INDTACPR) for System B. If these two conditions are met, the compression levels for System B are used. Otherwise, no session is established.
8. The inbound data compression level (INDTACPR) for System B must equal the outbound data compression level (OUTDTACPR) for System A. Conversely, the OUTDTACPR parameter for System B must match the INDTACPR parameter for System A. If they are equal, the session uses those compression levels. Otherwise, no session is established.

This table makes the following simplifications:

- DTACPR(*line-speed*) is not shown. When the line speed of the line is less than or equal to the value of DTACPR, the compression level is the same as when DTACPR is *REQUEST. When the line speed of the line is greater than the value of DTACPR, the compression level is the same as when DTACPR is *ALLOW.
- Both systems support APPC data compression. For any APPC system that does not support compression, the resulting compression level is the same as when DTACPR is *NONE.

***No Compression Sometimes Means No Session:*** When the DTACPR parameter is *REQUIRE, the INDTACPR and OUTDTACPR parameters specify the level of compression that is required for the inbound and outbound data. If that level of compression is not agreed to by the other system, the session is not started.

***Intermediate Node Compression Requests:*** When both systems allow compression but neither system requests compression, the data is compressed if an intermediate node requests compression. A system allows compression but does not request it when:

- DTACPR is *ALLOW
- DTACPR is *line-speed* and the line speed of the line is greater than the value of DTACPR.

***Data Compression Based on Line Speed:*** When the value of the DTACPR parameter is a line speed:

- Data compression is requested (the same as for *REQUEST) when the line speed is less than or equal to the value specified
- Data compression is allowed (the same as for *ALLOW) when the line speed is greater than the value specified

Whether or not the data is compressed depends on the DTACPR parameter of the partner system, see Table 3-2 on page 3-8.

***Determining the Compression Level:*** The compression level is specified on the inbound (INDTACPR) and outbound (OUTDTACPR) data compression parameters. The INDTACPR parameters of the systems are matched with the OUTDTACPR parameters of their partner system. The two systems negotiate to determine the compression levels for the session. The minimum compression levels of the matched INDTACPR and OUTDTACPR parameters are used. The compression levels from minimum to maximum are:

```
  Minimum                Maximum
None   RLE   LZ9   LZ10   LZ12
```

Table 3-3 shows the level of compression resulting from the settings shown for the INDTACPR or OUTDTACPR parameters.

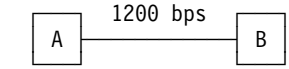
*Table 3-3 (Page 1 of 2). Minimum Compression Level Chart*

| | ***NONE** | ***RLE** | ***LZ9** | ***LZ10** | ***LZ12** |
|---|---|---|---|---|---|
| ***NONE** | None | None | None | None | None |

| Table 3-3 (Page 2 of 2). Minimum Compression Level Chart | | | | | |
|---|---|---|---|---|---|
| | ***NONE** | ***RLE** | ***LZ9** | ***LZ10** | ***LZ12** |
| ***RLE** | None | RLE | RLE | RLE | RLE |
| ***LZ9** | None | RLE | LZ9 | LZ9 | LZ9 |
| ***LZ10** | None | RLE | LZ9 | LZ10 | LZ10 |
| ***LZ12** | None | RLE | LZ9 | LZ10 | LZ12 |

**Data Compression Examples:** The following examples illustrate how some of the considerations for data compression may influence the configuration of APPC data compression. Use these examples to identify the considerations that are appropriate in your environment. Do not use the values shown without considering their effect in your environment.

***Example 1. Slow Line:*** System A and B are both configured to request compression.




```
Network              Network
Attributes           Attributes

 DTACPR=*REQUEST     DTACPR=*REQUEST

Mode                 Mode

 DTACPR=*NETATR      DTACPR=*NETATR
 INDTACPR=*LZ12      INDTACPR=*LZ10
 OUTDTACPR=*LZ12     OUTDTACPR=*LZ10
```

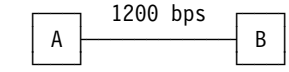
*Considerations*

- Slow line.
- Processor and storage resources are available. Otherwise, a lower compression level (LZ10, LZ9, or RLE) may be used.

*Resulting Compression Levels:* The compression level is negotiated and the minimum level is selected.

| | |
|---|---|
| A to B | LZ10 |
| B to A | LZ10 |

***Example 2. Slow Line, Another Solution:*** System A is configured to request compression and System B is configured to allow compression.




```
Mode                 Mode

 DTACPR=*REQUEST     DTACPR=*ALLOW
 INDTACPR=*LZ10      INDTACPR=*LZ12
 OUTDTACPR=*LZ9      OUTDTACPR=*RLE
```

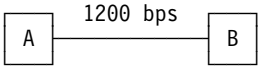
*Considerations*

- Slow line.
- Processor and storage resources are available. Otherwise, a lower compression level (LZ10, LZ9, or RLE) may be used.

*Resulting Compression Levels:* The compression level is negotiated and the minimum level is selected.

| | |
|---|---|
| A to B | LZ9 |
| B to A | RLE |

***Example 3. Slow Line, Yet Another Solution:*** System A is configured to request compression if the line speed between A and B is 2400 bps or less. It is configured to allow compression if the line speed between A and B is greater than 2400 bps. System B is configured to allow compression.




```
Mode                 Mode

 DTACPR=2400         DTACPR=*ALLOW
 INDTACPR=*LZ10      INDTACPR=*LZ12
 OUTDTACPR=*LZ10     OUTDTACPR=*LZ9
```

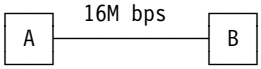
*Considerations*

- Slow line.
- Processor and storage resources are available. Otherwise, a lower compression level (LZ10, LZ9, or RLE) may be used.

*Resulting Compression Levels:* The compression level is negotiated and the minimum level is selected.

| | |
|---|---|
| A to B | LZ10 |
| B to A | LZ9 |

***Example 4. Fast Line:*** System A and B are both configured to allow compression.



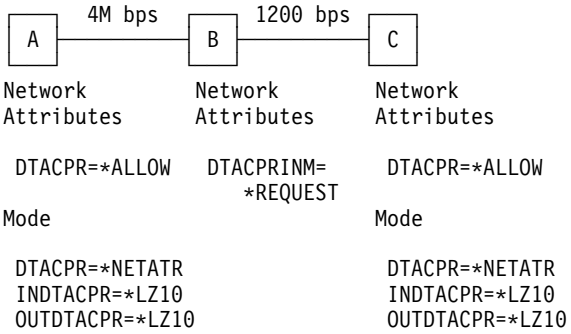

```
Mode                 Mode

 DTACPR=*ALLOW       DTACPR=*ALLOW
```

*Considerations*

- Fast line

*Resulting Compression Levels:* The data is not compressed because neither system requests compression.

***Example 5. One Fast Line and One Slow Line:*** System A and System C are both configured to allow compression. System B is configured to request compression for any sessions that use System B as an intermediate node.



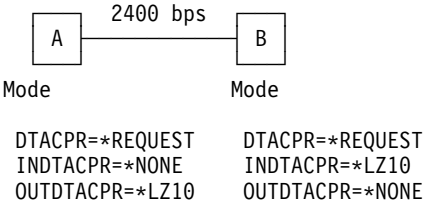

*Considerations*

- Fast line from System A to System B.
- Slow line from System B to System C.
- Processor and storage resources are available.

*Resulting Compression Levels*

| | |
|---|---|
| A to C | LZ10 |
| C to A | LZ10 |

***Example 6. Heavy Data Traffic in One Direction***



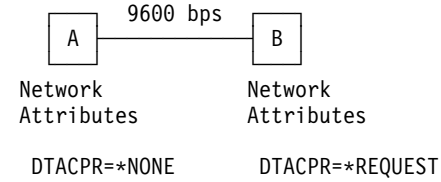

*Considerations*

- Slow line.
- More data is sent from System A to System B than from System B to System A. The level of compression specified on the OUTDTACPR parameter on System A maps to the level specified on the INDTACPR parameter on System B.
- Processor and storage resources are available.

*Resulting Compression Levels*

| | |
|---|---|
| A to B | LZ10 |
| B to A | None |

***Example 7. Processor Utilization Too High***



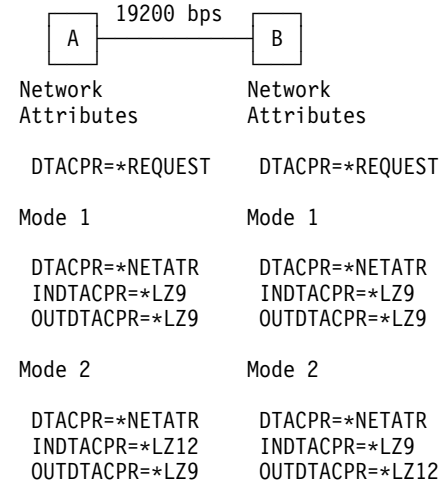

*Considerations*

- Slow line.
- System A does not allow compression because its processing unit is already busy.

*Resulting Compression Levels*

| | |
|---|---|
| A to B | None |
| B to A | None |

***Example 8. Specialized Mode***




*Considerations*

- 19200 bps line speed.
- Processor and storage resources are available.
- Off-shift large file transfers from System B to System A.

*Note:* Additional modes may be needed to handle unique compression needs.

*Resulting Compression Levels*

| | |
|---|---|
| A to B (Mode 1) | LZ9 |
| B to A (Mode 1) | LZ9 |
| A to B (Mode 2) | LZ9 |
| B to A (Mode 2) | LZ12 |

## When Do Changes Take Effect?

For APPC, changes to the mode description or network attributes do not take effect immediately. They take effect for sessions on a device when the device is varied off and then back on.

For APPN support, changes to the mode description or network attributes do not take effect immediately. They take effect for sessions on a device when all devices with the same remote location name, local location name, and remote network ID are varied off.

## How to Determine If a Session Uses Compression

For APPC, trace the communications line, which you can do using the Start System Service Tools (STRSST) command menu interface. For more information about the STRSST command, see the *CL Reference*.

For APPN support, display the unformatted BIND using the Display APPN Information (DSPAPPNINF) command.

In either case, the *SNA Formats* manual contains information to help you find the negotiated compression values.

# 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
- **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 nonsecure 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.

## 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 Table 3-4 for a list of the null password implementations for the AS/400 system, System/38, and System/36.

*Table 3-4. Null Password Implementations*

| System | Parameter | Null Value |
|---|---|---|
| AS/400 System | Remote Config. List Loc. Password or LOCPWD parameter | *NONE |
| System/38 | CRTDEVD SYSVLDPWD | *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 password can default to a value of null without reducing application program security.

## Validation Tables for Establishing a Session

Table 3-5 and Table 3-6 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), 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.

*Table 3-5 (Page 1 of 2). 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 |

*Table 3-5 (Page 2 of 2). 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)** |

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 without a password) are allowed.

4 Security is accepted but the already verified indicator is never sent.

*Table 3-6. 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 |

1 The information under each password location indicates the effect of each password location on session establishment (BIND).

2 Security accepted indicates that the incoming conversation-level security parameters (user ID and password, or user ID and already verified indicator without a password) are allowed.

## Failure to Establish a Session

With the addition of support for APPC password protection and two-phase commit there have been occurrences of failed session establishments (or BINDs) with non-OS/400 systems. There have also been failed BIND requests related to the V3R1 support of full-duplex for Anynet/400 Sockets over SNA, also with non-OS/400 systems. These failures are the result of incorrect support (by non-OS/400 systems) of the password protection, two-phase commit, and full-duplex bits in the BIND requests sent by OS/400.

## Examples of Failures

When the BIND negotiation fails (because of incorrect support of the LU 6.2 architecture by non-OS/400 systems), the following SNA sense codes may occur:

*Table 3-7. Sense Code Descriptions*

| Sense Code | Description | Comments. |
|---|---|---|
| X'080F 6051' | End User of LU Not Authorized. | The requesting end user or LU does not have the proper security authorization to access the requested resource.<br>6051 Access Security Information Invalid: The request specifies an Access Security Information field that is unacceptable to the receiver; for security reasons, no further detail on the error is provided. This sense data is sent in FMH-7 or UNBIND.<br>**Note:** X'080F' has many causes; X'6051' is just one example. You can look for the CPF1269 message by using the DSPLOG CL command. |
| X'0835 0007' | Invalid Parameter (with Pointer Only). | The request contained a fixed- or variable-length field whose contents are invalid or not supported by the NAU that received the request.<br>nnnn Bytes 2 and 3 contain a two-byte binary count that indexes (0-origin) the first byte of the fixed- or variable-length field having invalid contents.<br>X'0007' is referring to the Normal-flow send/receive mode selection. |
| X'0835 0024' | See previous sense code. | X'0024' is referring to the synchronization level. |
| X'086F 0000' | Length error. | A length field within a structure is invalid, or two or more length fields are incompatible. |
| X'1006 0006' | Required field or parameter is missing. | 0006 A required subfield of a control vector was omitted. |
| X'1008 6021' | Invalid FM Header. | The FM Header was not understood or translatable by the receiver, or an FM header was expected but not present. For LU 6.2, this sense code is sent in FMH-7 or UNBIND.<br>6021 Transaction Program Name Not Recognized: The FMH-5 Attach command specifies a transaction program name that the receiver does not recognize. This sense data is sent only in FMH-7. |
| X'1008 6034' | See previous sense code. | X'6034' is referring Conversation Type Mismatch: The FMH-5 Attach command specifies a conversation type that the receiver does not support for the specified transaction program. This sense data is sent only in FMH-7. |

If any of the previous sense codes are received as part of a failure to establish a session, the failure is probably related to incorrect support (by the remote non-OS/400 system) of:

- Password protection
- Two-phase commit
- Full-duplex

You should check to verify proper support of the BIND request by the non-OS/400 system that received the request.

**Note:** The previous sense code explanations are taken from the *SNA Formats*publication.

## Examples of Failures - by System

*Table 3-8. Sense Code Descriptions*

| System | Failure | Fix Available? |
|---|---|---|
| System/36 | Echoed the reserved bit for password substitution (password protection). | Yes, (System/36 PTF) |
| Novell Netware for SAA | Echoing the full-duplex bit. | Yes, contact Novell. |
| Novell Netware for SAA | Echoing the already verified bit. | Yes, contact Novell. |
| Novell Netware for SAA | Echoing the password substitution bit. | Yes, contact Novell. |
| SUNLINK for SUN | Did not expect user subfield 13 for password substitution. | Yes, contact SUN. |
| Pearl OEM "look alike" 5494 Controller | Fields are incompatible. | Aware of problem. |
| Idea OEM "look alike" 5494 Controller | Fields are incompatible. | Aware of problem. |

## Conversation Level Security

ICF supports three types of security that are specified on the SECURITY DDS keyword:

*Table 3-9. ICF Security Support*

| LU6.2 Architected Term | DDS Keyword |
|---|---|
| SECURITY(NONE) | no keyword or SECURITY(3 *NONE) |
| SECURITY(SAME) | SECURITY(3 *USER) |
| SECURITY(PGM) | SECURITY(1 profile 2 password 3 user ID) |

CPI-Communications supports four types of security that are specified on the Set_Conversation_Security_Type call:

*Table 3-10. CPI-Communications Security Support*

| LU6.2 Architected Term | CMSCST Value. |
|---|---|
| SECURITY(NONE) | CM_SECURITY_NONE |
| SECURITY(SAME) | CM_SECURITY_SAME (default value) |
| SECURITY(PGM) | CM_SECURITY_PROGRAM |
| SECURITY(PROGRAM_STRONG) | CM_SECURITY_PROGRAM_STRONG |

For the conversation level security implementation done by CICS/400, refer to the *CICS/400 Administration and Operations Guide* book. For sockets conversation level security implementation, refer to the *Sockets Programming* book.

## Implementations of LU6.2 Conversation Level Security

These descriptions assume that the remote system is the source of the conversation and the local system (an AS/400) is the target of the conversation.

**SECURITY(NONE):** SECURITY(NONE) conversations send no user ID, no password, and no profile from the remote (source) system. On AS/400s receiving SECURITY(NONE) program start requests, the DFTUSR parameter on the communications entry for the subsystem description (where the remote job runs) controls if the job starts.

The DFTUSR parameter can have these values:

***SYS**
This allows IBM-supplied programs (for example, SNADS, DDM, DRDA, display station pass-through) to run without sending security information. *SYS rejects programs that IBM does not supply (for example, user-written ICF or Common Programming Interface Communications applications).

***NONE**
All programs that do not send security information are rejected. This includes IBM-supplied and user written programs.

**user-profile-name**
All programs (IBM-supplied and others) that do not send security information will run under control of this user profile. All programs are treated the same. This is not desirable if you do not know what programs are being requested. This user profile should have minimal authority, and the user profile should not be specified on an *ALL communications entry. The user profile should only be specified on communications entries for specific locations - if it is used at all.

If there is no communications entry on any active subsystem then all attempts to start a SECURITY(NONE) conversation will fail. For more information on communications entries refer to the *Work Management* book.

**SECURITY(SAME):** SECURITY(SAME) conversations are dependent on local (target) system configurations for the degree of conversation level security. The SECURELOC (secure location) parameter on the APPC device description or APPN remote configuration list entry controls the level of security. The value is sent to the remote system when sessions are established. The value is used to determine how allocate or evoke requests should be built. The value applies only to conversations that are started with the SECURITY(SAME) level of security. The target system enforces this value.

SECURELOC has the following values:

***NO**

- The remote system is not a secure location.
- Security validation that is done by the remote system is not accepted. SECURITY(SAME) conver-

sations are treated as SECURITY(NONE) by the remote system.

- No security information is sent with allocate or evoke requests. This means that the local (target) system has no indication of who the user is on the remote system that is sending the program start request.
- The use of communications entries, as described previously, is applied to this situation.

***YES**

- The remote system is a secure location, and the local system accepts security validation that is done by remote systems.
- For SECURITY(SAME) conversations, the local system allows the remote system to verify user passwords and send an already verified indicator with the allocate or evoke requests.
- No password is sent. This means that any user with the same user profile name on both systems can run programs on the local system. If the user on the source system does not exist on the target system, the request is rejected.

  **Note:** You must trust the security/password checking on the remote system if you are using SECURELOC(*YES) since any user with the same user profile on each system can sign on and run jobs on your local system. QSECOFR is an example of a user profile that exists on all systems. Whoever can use the QSECOFR profile on the remote system can also use it on the local system if SECURELOC is *YES. This is a very powerful feature of SECURELOC(*YES).

***VFYENCPWD**

- The remote system is not a secure location. For SECURITY(SAME) conversations, the remote system is not allowed to send the already verified indicator.
- On the remote system, user IDs and passwords are retrieved from the security manager.
- Passwords are encrypted and sent with the user ID on allocate or evoke requests, to be verified by the local system.
- If the remote system does not support password protection, then session establishment is not allowed.
- For remote systems supporting password protection, but not supporting verification of encrypted passwords (VFYENCPWD), conversations are treated as SECURITY(NONE).
- The user profile names and passwords for users that are the same person on each system must be the same.
- The retrieval of the user ID and password is done by APPC and applications do not need to change to use this option.
- Using this option along with ensuring all user profiles and passwords are non-trivial significantly reduces the risk that someone can run jobs on the local system by pretending to be someone else (or being mistaken for someone else).

  **Note:** Generally, *VFYENCPWD allows you to trust some users but not all users on the remote system.

**SECURITY(PGM):** SECURITY(PGM) conversations send user ID, password, and profile that are based on what the application program specifies. The profile is optional on SECURITY(PGM) requests. Passwords may or may not be encrypted (known as protected passwords). For more information on protected passwords see “Password Protection” on page 3-18.

**SECURITY(PROGRAM_STRONG):** This is similar to SECURITY(PGM) with encrypted passwords. The difference is that an error is returned if the target system does not support password protection (cannot encrypt passwords). This is a good way of ensuring passwords are not intercepted on the communications line, something SECURITY(PGM) does not ensure.

## Enhanced SECURITY(SAME)

Enhanced SECURITY(SAME) on the OS/400 is used for CPI Communications SECURITY(SAME) conversations. Enhanced SECURITY(SAME) occurs when one of the following occurs:

- The target system specifies SECURELOC(*NO).
- Each system supports password protection.

For enhanced SECURITY(SAME):

- The remote system retrieves the user ID and password.
- The password is encrypted and sent along with the user ID on the allocate request.
- On the local system, the user ID must exist and the password must be the same on each system.

Normal SECURITY(SAME) would cause the conversation to be treated as SECURITY(NONE) because SECURELOC is *NO. Enhanced SECURITY(SAME) is similar to SECURITY(SAME) with the target having SECURELOC(*VFYENCPWD) described previously. An important difference is that enhanced SECURITY(SAME) is controlled on the source system, while the SECURELOC(*VFYENCPWD) is controlled on the target system.

Enhanced SECURITY(SAME) is enabled by setting the *conversation_security_type* to the CM_SECURITY_SAME value with the Set_Conversation_Security_Type call. This value is already the default value, if you do not explicitly set it with CMSCST then the SECURITY(SAME) conversations continue to be treated as SECURITY(NONE) when target systems specify SECURELOC(*NO).

If you use enhanced SECURITY(SAME) support:

- The user ID must explicitly exist on both systems.
- The password must be the same on both systems.

If these conditions are not met, the program start request is rejected with a security violation error. To avoid this error, do one of the following:

- Remove the Set_Conversation_Security_Type (CMSCST) call.
- Change the conversation security level to SECURITY(NONE) or SECURITY(PGM). For SECURITY(PGM) specify a valid user ID and password.
- Add the user ID or correct the password on the target system.
- Have the local system acknowledge that the source system is a secure location (SECURELOC(*YES)).

**Note:**

1. Enhanced SECURITY(SAME) is not used for ICF or CICS/400 SECURITY(SAME) conversations.
2. Enhanced SECURITY(SAME) is only used for CPI Communications when CM_SECURITY_SAME is set by the CMSCST call.

## Degrees of Conversation Level Security

The following table illustrates the degree of security target systems can expect with each of the conversation security levels:

*Table 3-11. Degrees of Conversation Level Security*

| Degree of Security | Conversation Level Security Used. |
|---|---|
| Failing Conversations | SECURITY(NONE) with no communications entry or *NONE in the communications entry<br>SECURITY(SAME) with SECURELOC(*NO) and no communications entry or *NONE in the communications entry |
| Least Secure Conversations | SECURITY(SAME) with SECURELOC(*YES)<br>SECURITY(NONE) with a communications entry allowing jobs to start with default user profiles<br>SECURITY(SAME) with SECURELOC(*NO) and a communications entry allowing jobs to start with default user profiles |
| More Secure Conversations | SECURITY(PGM) with password in the clear |
| Most Secure Conversations | SECURITY(PGM) with a protected (encrypted) password<br>Enhanced SECURITY(SAME)<br>SECURITY(SAME) with SECURELOC(*VFYENCPWD)<br>SECURITY(PROGRAM_STRONG) |

## Password Protection

With password protection support APPC substitutes a character string, called a **protected password**, for a user password when APPC starts a conversation. To use this support, the following conditions must be met:

- The systems of both partners must support password protection. This is done for OS/400 Version 3 Release 1 or later.
- The password must have been created on a system that runs OS/400 Version 2 Release 2 or later.
- The conversation level of security must be either SECURITY(PGM), SECURITY(PROGRAM_STRONG), enhanced SECURITY(SAME), or SECURITY(SAME) with the target having SECURELOC(*VFYENCPWD).

When password protection is available it reduces the chances that someone monitoring the communications line can discover the password of a User ID needed to run jobs on the system. SECURITY(PROGRAM_STRONG), enhanced SECURITY(SAME), and SECURITY(SAME) with SECURELOC(*VFYENCPWD) require password protection support to provide the function requested. SECURITY(PGM) uses password protection support if it is available but will work with or without it.

## System/38 and System/36 Secure Locations

Secure locations are defined on the System/38 by using the SECURELU parameter on the CRTDEVD command. The values for this parameter are *YES and *NO. The secure value is *YES. The nonsecure value is *NO.

For the System/36, secure locations are defined by using the 'Require User Password' option on SECEDIT COMM. The values are N and Y. The secure value is N. The nonsecure value is Y.

## System-Supplied Format Security

For system-supplied formats which use security information ($$EVOK, $$EVOKET, $$EVOKNI) the security information is specified in the data buffer of the source-side 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, 'blanks' operate the same as *USER
  - When specified for the password, 'blanks' operate the same as *NONE
- *NONE: No value is to be used
- *USER: The user ID of the currently signed on user

Table 3-12 shows the values that can be specified for the system-supplied formats security fields, and the resultant conversation security levels that are used.

*Table 3-12. Effects of System-supplied Format Security Fields on Conversation Level Security*

| | | **User ID Field** | | | |
|---|---|---|---|---|---|
| | | 'literal' | 'blanks' | *NONE | *USER |
| **Password Field** | 'literal' | SECURITY (PGM) | SECURITY (PGM) | SECURITY (PGM) | SECURITY (PGM) |
| | 'blanks' | SECURITY (PGM) | SECURITY (SAME) | SECURITY (NONE) | SECURITY (SAME) |
| | *NONE | SECURITY (PGM) | SECURITY (SAME) | SECURITY (NONE) | SECURITY (SAME) |
| | *USER | SECURITY (PGM) | SECURITY (PGM) | SECURITY (PGM) | SECURITY (PGM) |

## User IDs Used when the AS/400 System Is the Target System

Table 3-13 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 SECURELOC(*YES) is specified
- Security accepted and SECURELOC(*NO) is specified
- Security accepted and SECURELOC(*VFYENCPWD) is specified

*Table 3-13 (Page 1 of 2). Summary of User IDs Used when the AS/400 System is the Target System*

| **Session Establishment** | **SECURITY(NONE) Applications using Subsystem Default User ID** | **SECURITY(SAME) Applications using Source Sign-on User ID** | **SECURITY(PGM) or SECURITY(PROGRAM_STRONG) Applications using User ID and Password Sent in Program Start Request** |
|---|---|---|---|
| Security not accepted[1] | DDM[2] SNADS[3] AS/400 display station pass-through S/36 display station pass-through S/38 display station pass-through FTS[4] User-written program | Not applicable, security is not accepted | Not applicable, security is not accepted |
| Security accepted[1] and conversation is not affected by SECURELOC | AS/400 display station pass-through S/38 display station pass-through SNADS User-written program | Not applicable, all SECURITY(SAME) conversations are affected by SECURELOC | AS/400 display station pass-through FTS User-written program |

*Table 3-13 (Page 2 of 2). Summary of User IDs Used when the AS/400 System is the Target System*

| Session Establishment | SECURITY(NONE) Applications using Subsystem Default User ID | SECURITY(SAME) Applications using Source Sign-on User ID | SECURITY(PGM) or SECURITY(PROGRAM_STRONG) Applications using User ID and Password Sent in Program Start Request |
|---|---|---|---|
| Security accepted[1] and SECURELOC(*YES) | Not applicable, the value of SECURELOC does not affect SECURITY(NONE) conversations | AS/400 display station pass-through S/36 display station pass-through DDM FTS User-written program | Not applicable, the value of SECURELOC does not affect SECURITY(PGM) conversations |
| Security accepted[1] and SECURELOC(*NO) | Not applicable, the value of SECURELOC does not affect SECURITY(NONE) conversations | User-written program CPI Communications programs using enhanced SECURITY(SAME)[5] Note: All other SECURITY(SAME) conversations are treated as SECURITY(NONE) when SECURELOC(*NO) is specified | Not applicable, the value of SECURELOC does not affect SECURITY(PGM) conversations |
| Security accepted[1] and SECURELOC(*VFYENCPWD) | Not applicable, the value of SECURELOC does not affect SECURITY(NONE) conversations | AS/400 display station pass-through DDM FTS User-written program | Not applicable, the value of SECURELOC does not affect SECURITY(PGM) conversations |

**Notes:**

1. A security level (QSECURITY) system value of 10 is not secure. A value of 20 or greater is secure.
2. Distributed data management
3. SNA distribution services
4. File transfer support
5. If enhanced SECURITY(SAME) is used by a CPI Communications program, the user ID is sent with a protected password. All other applications will have SECURITY(SAME) treated as SECURITY(NONE).

## Converting User IDs and Passwords to Upper Case

APPC converts a lower case or mixed case user ID to upper case so the user ID is accepted by the AS/400. Passwords are also converted to upper case unless they are passwords substitutes, that is, encrypted.

This lower-to-upper case conversion supports AIX systems that connect to the AS/400 as a database server. For example, AIX users are often defined as lower case names and use lower case passwords.

AS/400 only accepts upper case entries for these fields. Because of this, lower case or mixed case user IDs or passwords that would have been rejected are now accepted.

## General Security Considerations

The following password considerations only apply if password protection is not active. When application program security is used, you should avoid sending and receiving passwords unnecessarily on start requests. The system starting 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.

## Incorrect Password Attempts

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 want 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.

## Password Expiration Management

Client users can change their expired AS/400 passwords when signing on to the AS/400 system. In order to do this, the client must support this function. The Client Access/400 clients support this function.

## 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.

# 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* book 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, which can include network server, network interface, 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 server, 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.

***NWS:** The network server and attached lines are varied.

***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 servers, the RANGE parameter is ignored; the attached lines are always varied. 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 DDI, frame relay, Ethernet, IDLC, token-ring, 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 configured in "Nonswitched Network without APPN Support—Configuration Example" on page D-3, the following example shows the VRYCFG commands that activate the line, controller, and devices on a non-switched line.

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 3-3 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 3-3 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 3-3 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 issued at any time. 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.

**CPLPNDRQS**
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)  CPLPNDRQS(*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 3-3 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.

**LCLLOCNAME**

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(APPCLOC1)  MODE(APPCMOD1)
  MAXSSN(10)
```

This command changes the maximum number of sessions allowed by the mode APPCMOD1 for remote location APPCLOC1 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 support) 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:   SYSNAM00
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

                                                                      Bottom
Press Enter to continue.

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:   SYSNAM00
Device  . . . . . . . . . :   LOSANGEL
Device status . . . . . . :   ACTIVE

Type options, press Enter.
  5=Display details

             Mode             ---------Conversations---------
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:   SYSNAM00
Device  . . . . . . . . . :   LOSANGEL
Device status . . . . . . :   ACTIVE

Type options, press Enter.
  5=Display details

             Mode             --Sessions--
Opt  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:   SYSNAM00
Device  . . . . . . . . . :   LOSANGEL
Device status . . . . . . :   ACTIVE

Type options, press Enter.
  5=Display details

             Mode             --Sessions--
Opt  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:   SYSNAM00
Device  . . . . . . . . . :   LOSANGEL
Device status . . . . . . :   ACTIVE

Type options, press Enter.
  5=Display details

             Mode             ---------Conversations---------
Opt  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:   SYSNAM00
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 . . . . . . . . . :     0         0         0         0
  Number for location . . . . . . . . :     0         0         0         0

Sessions:                                Total     Local     Remote
  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
```

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)
```

```
                         Display Mode Status
                                                          System:   SYSNAM00
Device  . . . . . . . . . . :   LOSANGEL
Device status . . . . . . . :   ACTIVE

Type options, press Enter.
  5=Display details

             Mode                ---------Conversations---------
Opt  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 Details of Mode Status
                                                          System:   SYSNAM00
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 . . . . . . . . . :     0         0         0         0
  Number for location . . . . . . . . :     0         0         0         0

Sessions:                                Total     Local     Remote
  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
```

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:   SYSNAM00
Device  . . . . . . . . . . :   LOSANGEL
Device status . . . . . . . :   ACTIVE

Type options, press Enter.
  5=Display details

             Mode                ---------Conversations---------
Opt  Mode    Status              Total  Source  Target  Detached
     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:   SYSNAM00
Mode/status . . . . . . . . . . . . . :   #BATCH       Ended
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 . . . . . . . . . . :       0
  Negotiated limits . . . . . . . . :       0         0         0
  Number for device . . . . . . . . :       1         1
  Number for location . . . . . . . :       1         1

                                                                      Bottom
Press Enter to continue.

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:   SYSNAM00
Device  . . . . . . . . . :   LOSANGEL
Device status . . . . . . :   ACTIVE

Type options, press Enter.
  5=Display details

                 Mode                ---------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 ILE C/400, ILE COBOL/400, FORTRAN/400, and ILE 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, Sense Data and Return Codes 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 Programming* book.

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.

**CRTICFF**
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.

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

**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.

***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 kept. 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 Programming* book 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 one of the following conditions is met.

- The send buffer is full.
- The program does an operation using an ALWWRT, CONFIRM, DETACH, FRCDTA, INVITE, or PRPCMT keyword.
- The program does a commit or rollback operation.

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 SYNLVL 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 Programming* book 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 operation (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 prestart 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 + length(parm1) + 4 +
  length(parm2) + ... + 4 + length(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 or two-phase commit 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. Two-phase commit processing is not allowed on SYNLVL(*CONFIRM) transactions. Refer to “Confirm Considerations” on page 7-3 for more information about confirm processing. Also, refer to the descriptions of the CONFIRM, RSPCONFIRM, and RCVCONFIRM keywords for more information on confirm processing.
- *COMMIT: Allows the programs to operate as described for the *CONFIRM value. Moreover, *COMMIT requires programs to use two-phase commit processing to protect their resources. Two-phase commit processing allows programs to synchronize updates to protected resources (such as databases). If necessary, updates can be rolled back, so that the resources remain synchronized. Refer to the descriptions of the PRPCMT, RCVROLLB, and RCVTKCMT keywords for more information on two-phase commit processing. The *Backup and Recovery* book has information about commitment control and the commit and rollback operations, which are an essential part of two-phase commit 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 or the TNSSYNLVL keyword.
3. The TNSSYNLVL keyword can be used with all synchronization levels. For more information about the TNSSYNLVL keyword, see “Transaction-Synchronization-Level Function” on page 5-6.

**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 3-12 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) or SYNLVL(*COMMIT) 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 5-9 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.

## Prepare-for-Commit Function

Your program uses the **prepare-for-commit** (PRPCMT) function to request one of its partners to prepare to commit its protected resources. The partner can respond with a commit, a rollback, or a FAIL operation. If the partner responds with a FAIL operation, the partner program is in control and can attempt to correct any errors that it detected.

The PRPCMT function contrasts with the commit operation in the following ways:

- PRPCMT only works with one conversation at a time. The commit operation attempts to commit all protected resources in the two-phase commit transaction program network.
- PRPCMT only prepares the remote protected resources to be committed. In other words, the remote resources have been locked and cannot be changed. They are in a state in which they can either be committed or rolled backed. Eventually, the remote resources are committed or rolled back depending on whether the rest of the two-phase commit transaction program network commits or rolls back its protected resources.

  The commit operation ends only after all remote protected resources in the two-phase commit transaction program network have either been committed or rolled back.
- PRPCMT allows the application program to attempt error recovery without rolling back the protected logical unit of work (LUW). When the application program issues a PRPCMT and the partner responds with a fail function, the PRPCMT function completes. The application program can then attempt error recovery, and issue the PRPCMT function again. The fail function is described in “Notifying the Remote Program of Problems by Using the Fail Function” on page 5-8.

  **Note:** The remote program is in send state after responding with the fail function. The local application program cannot issue the PRPCMT function again until the conversation states change.

  When the application program issues a commit operation and the partner responds with a fail function, the logical unit of work is rolled back.

An operation that includes the prepare-for-commit function does not complete until the remote program responds with a commit or rollback operation or a FAIL or EOS function.

After the PRPCMT function completes successfully, your program can do any one of the following.

- Use the commit operation to commit protected resources.
- Use the rollback operation to roll back the protected logical unit of work (LUW).
- Use the end-of-session function to end the attachment of the program to a session and roll back the protected LUW.

**Note:** The prepare-for-commit function only applies when SYNLVL(*COMMIT) 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 commit. 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 5-9 for more information.

The prepare-for-commit function causes any data currently held in the buffer to be sent, including any data on a write operation that specified the prepare-for-commit function. Refer to “Two-Phase Commit Considerations” on page 7-4 for additional information.

## Transaction-Synchronization-Level Function

Your program uses the **transaction-synchronization-level** (TNSSYNLVL) function to specify that synchronization for this transaction should be done at the level that the SYNLVL keyword specified on the evoke.

The TNSSYNLVL keyword can only be used if specified with one of the following keywords.

- ALWWRT
- DETACH
- INVITE

The following topics have more information about the transaction-synchronization-level function.

- “Allow-Write Function” on page 5-9
- “Ending a Transaction Using the Detach Function” on page 5-9
- “Invite Function” on page 5-7

## 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 (VARBUFMGT) function.

Using the VARBUFMGT 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 5-1This 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                    VARBUFMGT
  REC1           30
R EXAMPLE2                    VARBUFMGT
                              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).
```

*Figure 5-2 (Part 1 of 3). COBOL/400 Example Program for Sending Multiple Fixed-Length Records*

```
 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 VARBUFMGT 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.
```

*Figure 5-2 (Part 2 of 3). COBOL/400 Example Program for Sending Multiple Fixed-Length Records*

```
 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 VARBUFMGT 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-O.
     MOVE 20 TO VLEN OF EXAMPLE2-O.
     WRITE ICFREC FORMAT IS "EXAMPLE2"
         TERMINAL IS PGM-DEV-NME.
```

*Figure 5-2 (Part 3 of 3). 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 (VARBUFMGT) 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.

When your application program specifies the TNSSYNLVL keyword with the invite function, the additional function performed depends on the synchronization level of the conversation. The TNSSYNLVL keyword can be specified with the invite function only if the synchronization level is *NONE or *CONFIRM. Table 5-1 shows the details.

*Table 5-1. TNSSYNLVL Function with Invite*

| Synchronization Level | Function |
|---|---|
| *NONE | The force-data function is performed in addition to the invite function. |
| *CONFIRM | The confirm function is performed in addition to the invite function. |
| *COMMIT | Not allowed |

## 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 on page 5-8 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 5-6.

```
     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 function 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 Programming* book 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.

When a fail function is the response to a commit operation, the system rolls back the protected LUW on the side that issued the commit operation. The side that issued the fail function must do a rollback operation after the request to roll back is received from the partner. When a fail function is the response to a PRPCMT function, APPC does *not* roll back the protected LUW. Since APPC does not do a rollback for PRPCMT, the application program can try to correct the problem.

## Using Additional Functions and Operations

APPC supports the additional functions and operation listed below. Refer to the *ICF Programming* book 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.

When your application program specifies the TNSSYNLVL keyword with the ALWWRT keyword, the additional function performed depends on the synchronization level of the conversation. Table 5-2 shows the details.

*Table 5-2. TNSSYNLVL Function with ALWWRT*

| Synchronization Level | Function |
|---|---|
| *NONE | The force-data function is performed in addition to the allow-write function. |
| *CONFIRM | The confirm function is performed in addition to the allow-write function. |
| *COMMIT | The conversation enters defer receive state until your application program issues a commit operation, a force-data function, or a confirm function. Once the commit operation, force-data function, or confirm function completes successfully; the conversation is in receive state. |

### 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 “GET_ATTRIBUTES” on page C-11 and to the *ICF Programming* book 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.

## Using the Detach Function When the Synchronization Level is None

When the synchronization level is none and the detach and transaction-synchronization-level functions are used together, force-data and detach functions are performed.

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

## Using the Detach Function When the Synchronization Level is Confirm

When a detach function and a confirm function or transaction-synchronization-level function are used together, a confirm function is performed. If the remote program responds positively, the detach function is performed. If the remote program responds negatively, or has already sent a negative response, the transaction may not end immediately. 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.

To respond positively to the detach function with a confirm or transaction-synchronization-level function, the remote program must use the respond-to-confirm function.

To respond negatively to the detach function with a confirm or transaction-synchronization-level function, the remote program should use the fail function.

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

## Using the Detach Function When the Synchronization Level is Commit

For two-phase commit processing, the detach function must be accompanied by the transaction-synchronization-level function. The transaction does not end until your program issues a commit operation, and the commit operation completes successfully. If the commit operation fails, the following is done.

- The logical unit of work is rolled back.
- The transaction is not ended.
- The conversation state is returned to what it was at the last commit boundary.

## Using the Detach Function From a Target Program

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 your program is the source of a conversation with a synchronization level of commit, the release operation causes a rollback to occur unless the release comes at a commit boundary.
- 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.
3. For two-phase commit processing, the EOS causes a rollback to occur.

## 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.

For two-phase commit processing, the close operation causes a rollback to occur if the session is ended.

## 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, Sense Data and Return Codes.

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 the following major return codes along with the appropriate minor return codes.

00 user data received

02 end job or end subsystem in progress

03 no data received

The appropriate minor return codes are 00, 04, 06, 13, 14, 44, and 58.

If the return code is 0358, the conversation is not in send state until after a successful commit operation.

## 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, 46, or 59.

If the return code is 0359, the conversation is not detached until after a successful commit operation.

## Receive-Control-Data

Your program uses the **receive-control-data** (RCVCTLDTA) response indicator to receive an indication that the record 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.

## Receive-Rollback

Your program uses the **receive-rollback** (RCVROLLB) response indicator to receive an indication of one of the following conditions.

- The remote program sent a ROLLBACK. This indicates that the remote program expects your program to rollback its protected resources.
- The protected LUW entered the rollback required state.

Your program must respond with a rollback operation. Your program can only get this response indicator if it has a conversation with a synchronization level of commit.

This response indicator can be received with the following return codes.

- 0054
- 0254
- 80F9, 80FA, 80FB
- 81F0, 81F1, 81F2, 81F3, 81F4, 81F5
- 83FB, 83FC, 83FD, 83FE, 83FF

## Receive-Take-Commit

Your program uses the **receive-take-commit** (RCVTKCMT) response indicator to receive an indication that the remote program sent a PRPCMT function or a commit operation. This indicates that the remote program expects your program to determine if it can commit its protected resources. Your program must either do a commit or rollback operation or a FAIL or EOS function. Your program can only get this response indicator if it has a conversation with a synchronization level of commit.

This response indicator can be received with major return codes 02 (end job or end subsystem in progress) or 03 (no data received). The major return code can be accompanied by a minor return code of 57, 58, or 59.

# 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 Support* book 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, Sense Data and Return Codes 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.

*Table 5-3 (Page 1 of 2). Mapping between ICF Operations or Functions and LU Type 6.2 Verbs*

| LU 6.2 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 |

*Table 5-3 (Page 2 of 2). Mapping between ICF Operations or Functions and LU Type 6.2 Verbs*

| LU 6.2 Mapped or Basic Conversation Verbs | ICF Operation or Function | DDS Keyword | System- Supplied Format |
|---|---|---|---|
| MC_DEALLOCATE or DEALLOCATE | Write operation with detach function, and, depending on the type of program, one of the following:<br>• For a source program, an end-of-session function, release, or close operation must follow the detach.<br>• For a target program, a close operation should follow the detach. | DETACH EOS | $$SENDET<br>$$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 | $$SEND with 0 length |
| MC_PREPARE_FOR_SYNCPT or PREPARE_FOR_SYNCPT | Prepare-for-commit function | PRPCMT | No equivalent |
| MC_PREPARE_TO_RECEIVE or PREPARE_TO_RECEIVE | Allow-write or invite function | ALWWRT or INVITE | $$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 | $$RCD<br>**Note:** This also causes an invite to be issued. |
| SEND_DATA | Write operation of a format with a data length greater than 2 | Not applicable | $$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 |
| MC_TEST or TEST | 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.

*Table 5-4 (Page 1 of 3). Mappings for Example ICF Operations and Functions*

| ICF Operations/Functions | LU Type 6.2 Verbs—Basic or Mapped |
|---|---|
| An acquire operation followed by:<br>`EVOKE([library-name/]program-name)`<br>**Note:** It is the combination of an acquire operation and evoke function that corresponds to [MC_]ALLOCATE. | Corresponds to:<br>`[MC_]ALLOCATE with TPN(program-name) followed by an [MC_]FLUSH` |
| With an evoke function, you can also specify any of the following: | |
| `[parameter-1...[parameter-255]]` | `[PIP(YES(variable1 variable2...variablen))]` |
| `SECURITY(3 *USER) or`<br>`SECURITY([1 profile ID] [2 password] [3 userid])` | `SECURITY(SAME) or`<br>`SECURITY(PGM)` |
| `SYNLVL(*NONE)` | `SYNC_LEVEL(NONE)` |
| `DFREVOKE` | `Do not follow the [MC_]ALLOCATE with an [MC_]FLUSH` |
| `SYNLVL(*CONFIRM) or`<br>`  SYNLVL(*COMMIT)` | `SYNC_LEVEL(CONFIRM) or`<br>`  SYNC_LEVEL(SYNCPT)` |
| ⋮ | ⋮ |
| `Write (data length > 0) with CONFIRM` | `[MC_]SEND_DATA followed by [MC_]CONFIRM` |
| If your program specifies SYNLVL(*CONFIRM) and the CONFIRM DDS keyword, the remote program must then issue a: | Corresponds to: |
| `RSPCONFIRM (positive response)`<br>`  or`<br>`FAIL (negative response)` | `[MC_]CONFIRMED`<br>`  or`<br>`MC_]SEND_ERROR` |
| If the remote program issues a<br>`Get-attributes operation`,<br>it can determine the SYNLVL established by the source program. | `[MC_]GET_ATTRIBUTES` |
| Your program issues a: | Corresponds to: |
| `FRCDTA` | `[MC_]FLUSH` |
| Your program issues a write with `FMTNAME` to send the record format name along with data to the remote system.<br>**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. | Corresponds to:<br>`MC_SEND_DATA with`<br>`  [MAP_NAME(YES(map-name))]` |
| Your program uses a:<br>`Write operation`<br>to send data to the remote system, and then uses a:<br>`Read operation`<br>to receive data from a specific program device (for example, PGMDEVA) and waits to receive data before continuing | Corresponds to:<br>`[MC_]SEND_DATA`<br><br>`[MC_]RECEIVE_AND_WAIT` |
| Your program uses a:<br>`Write operation with a CTLDTA keyword`<br>If your program issues a:<br>`Write without data and INVITE`<br>requesting data from the remote program to send data for a program device (for example, PGMDEVA), your program receives control without waiting for input. | Corresponds to:<br>`[MC_]SEND_DATA with USER_CONTROL_DATA`<br>Corresponds to:<br>`[MC_]POST_ON_RECEIPT or [MC_]PREPARE_TO_RECEIVE followed by [MC_]POST_ON_RECEIPT, if the program is in send state` |
| You can also specify the following with the invite function:<br>`TNSSYNLVL`<br>unless your program specified SYNLVL(*COMMIT). | `TYPE(SYNC_LEVEL) specified with the [MC_]PREPARE_TO_RECEIVE` |
| Later, your program can issue a<br>`Read-from-invited-program-devices operation.`<br>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. | `WAIT` |
| Your program issues a:<br>`Write without data and RQSWRT`<br>to indicate that it wants to send data to the remote program rather than continue receiving data. | Corresponds to:<br>`[MC_]REQUEST_TO_SEND` |
| Your program issues a:<br>`Write with data and ALWWRT`<br>to inform the remote program that your program is done sending data and is ready to receive. | Corresponds to:<br>`[MC_]SEND_DATA followed by`<br>`  [MC_]PREPARE_TO_RECEIVE` |

*Table 5-4 (Page 2 of 3). Mappings for Example ICF Operations and Functions*

| ICF Operations/Functions | LU Type 6.2 Verbs—Basic or Mapped |
|---|---|
| Your program issues a: | Corresponds to: |
| `Write with data, ALWWRT, and TNSSYNLVL` | `[MC_]SEND_DATA followed by [MC_]PREPARE_TO_RECEIVE with TYPE(SYNC_LEVEL)` |
| to inform the remote program that your program is done sending data and is ready to receive. | |
| If your program specified SYNLVL(*COMMIT), your program issues a: | |
| `Commit operation` | `SYNCPT` |
| to commit the changes. | |
| Your program issues a: | Corresponds to: |
| `Write without data and ALWWRT` | `[MC_]PREPARE_TO_RECEIVE` |
| to inform the remote program that your program is done sending data and is ready to receive. | |
| Your program issues a: | Corresponds to: |
| `Write without data, ALWWRT, and TNSSYNLVL` | `[MC_]PREPARE_TO_RECEIVE with TYPE(SYNC_LEVEL)` |
| to inform the remote program that your program is done sending data and is ready to receive. | |
| If your program specified SYNLVL(*COMMIT), your program issues a: | |
| `Commit operation` | `SYNCPT` |
| to commit the changes. | |
| Your program issues a: | Corresponds to: |
| `Write without data and PRPCMT` | `[MC_]PREPARE_FOR_SYNCPT` |
| To end a session, a source program issues: | Corresponds to: |
| `Write with DETACH, followed by EOS or close` | `[MC_]DEALLOCATE with TYPE(FLUSH)` |
| **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. | |
| If your program issues: | Corresponds to: |
| `DETACH with CONFIRM, followed by EOS or close,` | `[MC_]DEALLOCATE with TYPE(CONFIRM)` |
| the transaction does not end if the remote program sends a | |
| `FAIL (negative acknowledgement)` | `[MC_]SEND_ERROR` |
| Otherwise, the transaction ends normally when the remote program issues a: | |
| `Write without data and RSPCONFIRM` | `[MC_]CONFIRMED` |
| If your program issues: | Corresponds to: |
| `DETACH with TNSSYNLVL` | `[MC_]DEALLOCATE with TYPE(SYNC_LEVEL)` |
| If your program specified SYNLVL(*COMMIT), the transaction does not end until your program issues a: | |
| `Commit operation` | `SYNCPT` |
| to commit the changes. | |
| If a program receives the following detach and commit indications for a protected conversation | Corresponds to: |
| `RCVDETACH and RCVTKCMT` | `WHAT_RECEIVED(TAKE_SYNCPT_DEALLOCATE) on an appropriate LU 6.2 operation` |
| (or it receives return codes 0259 or 0359), it should commit changes using: | |
| `Commit operation` | `SYNCPT` |
| If the program is a target program, started by a program start request, and it receives a detach indication for an unprotected conversation using: | Corresponds to: |
| `RCVDETACH` | `RETURN_CODE(DEALLOCATE_NORMAL) on an appropriate LU 6.2 operation` |
| (or it receives return codes 0008, 0208, or 0308), it should end the session using: | |
| `EOS` | `[MC_]DEALLOCATE with TYPE(LOCAL)` |
| If the program is a source program, and it receives a detach indication for an unprotected conversation using: | Corresponds to: |
| `RCVDETACH` | `RETURN_CODE(DEALLOCATE_NORMAL) on an appropriate LU 6.2 operation` |

*Table 5-4 (Page 3 of 3). Mappings for Example ICF Operations and Functions*

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

## 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

Figure 5-4 on page 5-18 shows the flow for the exchange of information that occurs in an inquiry application using ICF operations and functions.

The steps shown in Figure 5-4 are described below.

*Table 5-5 (Page 1 of 2). 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.<br>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. |

*Table 5-5 (Page 2 of 2). Description of Flow Diagram*

| Step | Description |
|---|---|
| 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. |

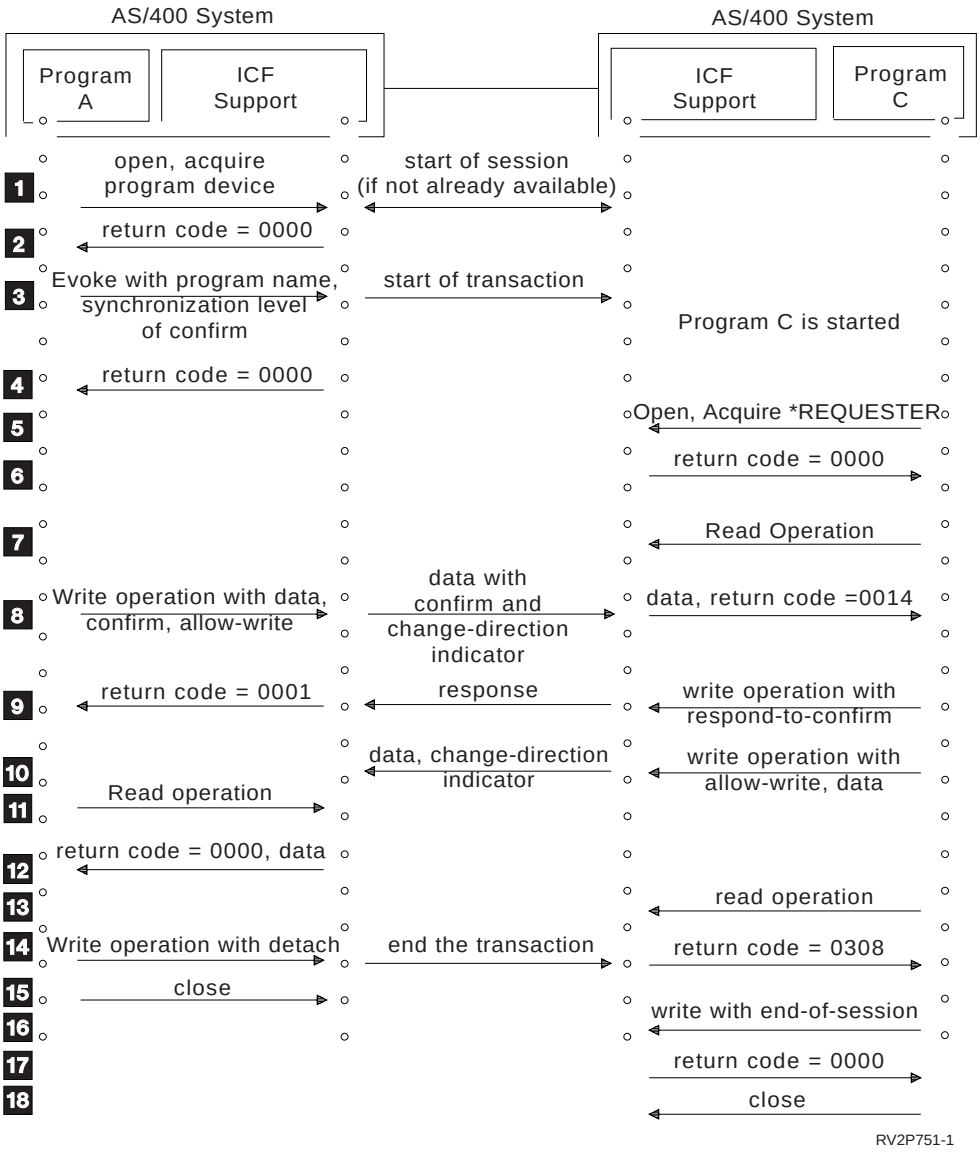



*Figure 5-4. Data Flow Using ICF Operations and Functions*

## Flow Diagram for Inquiry Applications Using LU Type 6.2 Verbs

Figure 5-5 on page 5-20 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 5-17.

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

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

*Table 5-6. 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. |
| 2 | No errors occurred on the MC_ALLOCATE, so the RETURN_CODE is set to OK. A *resource_ID* 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 *resource_ID* 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 | 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 *resource_ID* 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.<br><br>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.<br><br>**Note:** The AS/400 system, for performance reasons, returns multiple values for WHAT_RECEIVED. |
| 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 *resource_ID* 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. |

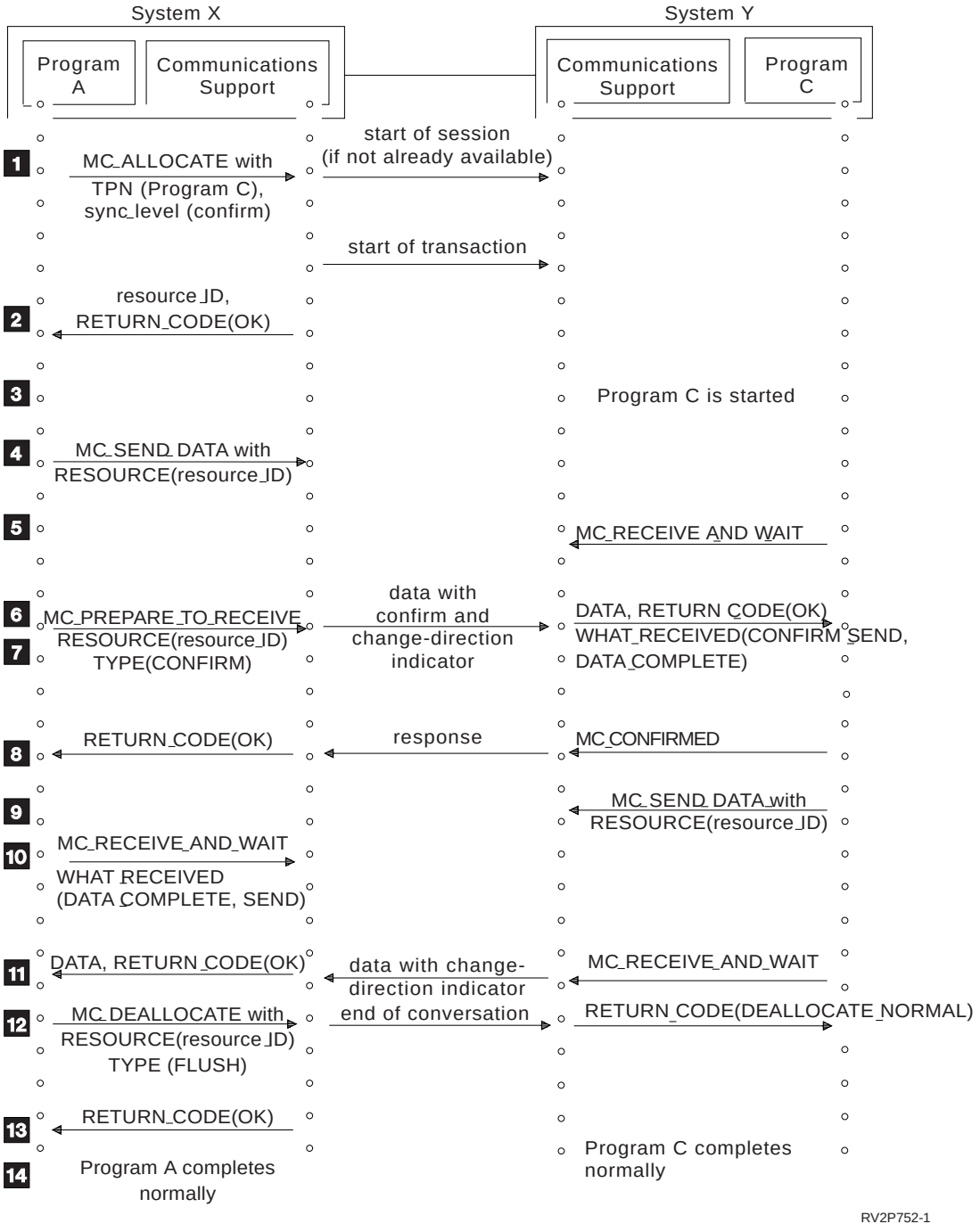



*Figure 5-5. Data Flow Using LU Type 6.2 Verbs*

# 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 list of the AS/400 CPI Communications calls, which allow a CPI Communications program to communicate with its partner program.
- A mapping of CPI Communications calls to ICF operations and functions.
- An example of a data flow using AS/400 CPI Communications.
- A list 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 full APPN support while using CPI Communications. Refer to “Using the Location Parameters” on page 3-3 for a detailed description of how the AS/400 system uses these parameters.

Table 6-1 describes the information contained in the communications side information object and maps it to the CPI Communications counterparts.

*Table 6-1 (Page 1 of 2). 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 *partner_LU_name*, which is defined by the CPI Communications architecture as a required characteristic for the communications side information. A fully-qualified *partner_LU_name* 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 *mode_name* characteristic.<br><br>To use a *mode_name* of eight blank characters, you must use the special value of BLANK. The AS/400 system does not support sending a *mode_name* of 'BLANKbbb' to a remote system. |

*Table 6-1 (Page 2 of 2). Description of Communications Side Information Object*

| AS/400 System Parameters | Description |
|---|---|
| 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 *TP_name*, 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 3-3 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:

- ILE C/400
- ILE COBOL/400
- Cross System Product (CSP) (implements the Application Generator common programming interface)
- FORTRAN/400
- REXX/400
- ILE 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:

**ILE C/400**

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

**ILE 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/400**

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

**ILE 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 Table 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.

*Table 6-2 (Page 1 of 3). 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 |
| CMCNVI | Convert_Incoming | Used by a program to convert from EBCDIC to the local character codes. No conversion is done when called on an AS/400 system. |
| CMCNVO | Convert_Outgoing | Used by a program to convert from the local character codes to EBCDIC. No conversion is done when called on an AS/400 system. |
| CMDEAL | Deallocate | Used by a program to end a conversation |

*Table 6-2 (Page 2 of 3). Supported CPI Communications Program Calls*

| Call Name | Pseudonym | Description |
|---|---|---|
| CMECS | Extract_Conversation_State | Used by a program to view the current *conversation_state* conversation characteristic |
| CMECT | Extract_Conversation_Type | Used by a program to view the current *conversation_type* conversation characteristic |
| CMEMBS | Extract_Maximum_Buffer_Size | Used by a program to view the maximum buffer size supported by the system. |
| CMEMN | Extract_Mode_Name | Used by a program to view the current *mode_name* conversation characteristic |
| CMEPLN | Extract_Partner_LU_Name | Used by a program to view the current *partner_LU_name* conversation characteristic |
| CMESL | Extract_Sync_Level | Used by a program to view the current *sync_level* conversation characteristic |
| CMESUI | Extract_Security_User_ID | Used by a program to view the current *security_user_ID* 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 |
| CMSCSP | Set_Conversation_Security_ Password | Used by a program to set the *security_password* and the *security_password_length* conversation characteristics |
| CMSCST | Set_Conversation_Security_ Type | Used by a program to set the *conversation_security_type* conversation characteristic |
| CMSCSU | Set_Conversation_Security_ User_ID | Used by a program to set the *security_user_ID* and the *security_user_ID_length* conversation characteristics |
| CMSCT | Set_Conversation_Type | Used by a program to set the *conversation_type* conversation characteristic |
| CMSDT | Set_Deallocate_Type | Used by a program to set the *deallocate_type* conversation characteristic |
| CMSED | Set_Error_Direction | Used by a program to set the *error_direction* conversation characteristic |
| CMSEND | 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 *fill* conversation characteristic |
| CMSLD | Set_Log_Data | Used by a program to set the *log_data* conversation characteristic |
| CMSMN | Set_Mode_Name | Used by a program to set the *mode_name* conversation characteristic |
| CMSPLN | Set_Partner_LU_Name | Used by a program to set the *partner_LU_name* conversation characteristic |
| CMSPTR | Set_Prepare_To_Receive_Type | Used by a program to set the *prepare_to_receive_type* conversation characteristic |
| CMSRC | Set_Return_Control | Used by a program to set the *return_control* conversation characteristic |
| CMSRT | Set_Receive_Type | Used by a program to set the *receive_type* conversation characteristic |

Table 6-2 (Page 3 of 3). Supported CPI Communications Program Calls

| Call Name | Pseudonym | Description |
|---|---|---|
| CMSSL | Set_Sync_Level | Used by a program to set the *sync_level* conversation characteristic |
| CMSST | Set_Send_Type | Used by a program to set the *send_type* conversation characteristic |
| CMSTPN | Set_TP_Name | Used by a program to set the *TP_name* 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. Table 6-3 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.

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

| Language | Member Name | Source Physical File | Library |
|---|---|---|---|
| ILE C/400 | CMC | H | QSYSINC |
| COBOL/400 | CMCOBOL | QLBLSRC | QSYSINC |
| ILE COBOL/400 | CMCOBOL | QCBLLESRC | QSYSINC |
| FORTRAN/400 | CMFORTRN | QIFOINC | QFTN |
| RPG/400 | CMRPG | QRPGSRC | QSYSINC |
| ILE RPG/400 | CMRPG | QRPGLESRC | QSYSINC |

**Notes:**

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

## Mapping of CPI Communications Calls to ICF Operations and Functions

Table 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.

Table 6-4 (Page 1 of 3). Mapping of CPI Communications Calls to ICF Operations and Functions

| Program Call and 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 |

*Table 6-4 (Page 2 of 3). Mapping of CPI Communications Calls to ICF Operations and Functions*

| Program Call and Pseudonym | ICF Operation or Function | DDS Keyword | System-Supplied Format |
|---|---|---|---|
| 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 |
| CMCNVI Convert_Incoming | No equivalent | No equivalent | No equivalent |
| CMCNVO Convert_Outgoing | No equivalent | No equivalent | No equivalent |
| CMDEAL Deallocate | Write operation with detach function, and, depending on the program, one of the following:<br>• For a source program, an end-of-session function, a release, or a close operation can follow the detach.<br>• 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 |
| CMEMBS Extract_Maximum_Buffer_Size | No equivalent | No equivalent | No equivalent |
| CMECS Extract_Conversation_State | Get-attributes operation | Not applicable | Not applicable |
| 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 |
| CMESUI Extract_Security_User_ID | 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<br>**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<br>**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 |
| CMSCSP Set_Conversation_Security_Password | Evoke function with security password | SECURITY(2 '*password*') | $$EVOKNI, $$EVOK, or $$EVOKET with password |

*Table 6-4 (Page 3 of 3). Mapping of CPI Communications Calls to ICF Operations and Functions*

| Program Call and Pseudonym | ICF Operation or Function | DDS Keyword | System-Supplied Format |
|---|---|---|---|
| CMSCST Set_Conversation_ Security_Type | Implicit in the security information (or lack thereof) supplied with an evoke function | SECURITY | Implicit in the security information (or lack thereof) supplied with an evoke format |
| CMSCSU Set_Conversation_ Security_User_ID | Evoke function with security user ID | SECURITY(3 *user*) | $$EVOKNI, $$EVOK, or $$EVOKET with user ID |
| CMSSL Set_Sync_Level | Evoke function with synchronization level | SYNLVL | No equivalent |
| 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 |

## ICF to CPI Communications—Examples

Table 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 DDS information contained in Appendix A.

**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.

*Table 6-5 (Page 1 of 3). Mappings for Example ICF Operations and Functions*

| ICF Operations/Functions | CPI Communications Call |
|---|---|
| An acquire operation followed by:<br>`EVOKE([library-name/]program-name)`<br><br>**Note:** It is the combination of an acquire operation and evoke function that corresponds to Allocate (CMALLC). | Corresponds to<br>`CALL CMALLC(conversation_ID)` |
| With an evoke function, you can also specify any of the following:<br>`[parameter-1...[parameter-255]]` | CPI Communications does not support sending PIP data. |
| `SECURITY(3 *USER) or`<br>`SECURITY([1 profile ID] [2 password] [3 userid])` | `CALL CMSCST(CM_SECURITY_SAME) or`<br>`CALL CMSCST(CM_SECURITY_PROGRAM)`<br>`CALL CMSCSP(security_password)`<br>`CALL CMSCSU(security_user_ID)` |
| `SYNLVL(*NONE) or`<br>`SYNLVL(*CONFIRM) or`<br>`SYNLVL(*COMMIT)` | `CALL CMSSL(CM_NONE) or`<br>`CALL CMSSL(CM_CONFIRM) or`<br>`CALL CMSSL(CM_SYNC_POINT)` (CMSCST, CMSCSP, CMSCSU, and CMSSL must all be called before the CMALLC call.) |
| ⋮ | ⋮ |
| `Write (data length > 0) with CONFIRM`<br>If your program specifies SYNLVL(*CONFIRM) and later uses the CONFIRM DDS keyword, the remote program must then issue a<br>`RSPCONFIRM (positive response)`<br>or<br>`FAIL (negative response)` | `CALL CMSST(CM_SEND_AND_CONFIRM) followed by CALL CMSEND`<br>Corresponds to<br>`CALL CMCFMD`<br>or<br>`CALL CMSERR or CALL CMSED followed by CALL CMSERR when in send-pending state` |
| If the remote program issues a<br>`Get-attributes operation`,<br>it can determine the SYNLVL established by the source program. | `CALL CMESL` |
| Your program issues a<br>`FRCDTA` | Corresponds to<br>`CALL CMFLUS` |

*Table 6-5 (Page 2 of 3). Mappings for Example ICF Operations and Functions*

| ICF Operations/Functions | CPI Communications Call |
|---|---|
| Your program uses a<br>`Write operation`<br>to send data to the remote system, and then uses a<br>`Read operation`<br>to receive data from a specific program device (for example, PGMDEVA) and waits to receive data before continuing. | Corresponds to<br>`CALL CMSEND`<br><br>`CALL CMRCV` |
| If your program issues an:<br>`INVITE`<br>requesting data from the remote program to send data for a program device (for example, PGMDEVA), your program receives control without waiting for input.<br><br>You can also specify the following with the invite function:<br>`TNSSYNLVL`<br>unless your program specified SYNLVL(*COMMIT). | Corresponds to<br>`CALL CMSPTR(CM_PREP_TO_RECEIVE_FLUSH) followed by CALL CMPTR`<br>**Note:** CPI Communications does not support the implied LU 6.2 [MC]_POST_ON_RECEIPT that is done for the ICF invite function.<br><br>`CALL CMSPTR(CM_PREP_TO_RECEIVE_SYNC_LEVEL) instead of CALL CMSPTR(CM_PREP_TO_RECEIVE_FLUSH)` |
| Later, your program can issue a<br>`Read-from-invited-program-devices operation`<br>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. | `No CPI Communications equivalent.`<br>However, you can issue a CMRCV with the *receive_type* 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. |
| Your program issues:<br>`RQSWRT`<br>to indicate that it wants to send data to the remote program rather than continue receiving data. | Corresponds to<br>`CALL CMRTS` |
| Your program issues a:<br>`Write with ALWWRT`<br>to inform the remote program that your program is done sending data and is ready to receive. | Corresponds to<br>`CALL CMSPTR(CM_PREP_TO_RECEIVE_FLUSH) or CALL CMSST(CM_SEND_AND_PREP_TO_RECEIVE) and CALL CMSEND`<br>if data is sent with a change-direction indicator |
| Your program issues a:<br>`Write with ALWWRT and TNSSYNLVL`<br>to inform the remote program that your program is done sending data and is ready to receive. | Corresponds to<br>`CALL CMSPTR(CM_PREP_TO_RECEIVE_SYNC_LEVEL) and CALL CMSEND`<br>if data is sent with a change-direction indicator |
| If your program specified SYNLVL(*COMMIT), your program issues a:<br>`Commit operation`<br>to commit the changes. | `Commit operation` |
| Your program issues a:<br>`Write with ALWWRT`<br>to inform the remote program that your program is done sending data and is ready to receive. | Corresponds to<br>`CALL CMSPTR(CM_PREP_TO_RECEIVE_FLUSH) and CALL CMPTR`<br>if no data is sent |
| Your program issues a:<br>`Write with ALWWRT and TNSSYNLVL`<br>to inform the remote program that your program is done sending data and is ready to receive. | Corresponds to<br>`CALL CMSPTR(CM_PREP_TO_RECEIVE_SYNC_LEVEL) and CALL CMPTR`<br>if no data is sent |
| If your program specified SYNLVL(*COMMIT), your program issues a:<br>`Commit operation`<br>to commit the changes. | `Commit operation` |
| Your program issues a:<br>`Write without data and PRPCMT` | Corresponds to:<br>`No CPI Communications equivalent.`<br>**Note:** Commit and rollback operations can be performed. |
| To end a session, a source program issues:<br>`Write with DETACH, followed by EOS or close`<br>**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. | Corresponds to<br>`CALL CMSDT(CM_DEALLOCATE_FLUSH)`<br>`CALL CMSST(CM_SEND_AND_DEALLOCATE), followed by CALL CMSEND`<br>or<br>`CALL CMSDT(CM_DEALLOCATE_FLUSH) followed by CALL CMDEAL` |

*Table 6-5 (Page 3 of 3). Mappings for Example ICF Operations and Functions*

| ICF Operations/Functions | CPI Communications Call |
|---|---|
| If your program issues:<br>`DETACH with CONFIRM, followed by EOS or close,`<br>the transaction does not end if the remote program sends a<br>`FAIL (negative acknowledgement)`<br>**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. | Corresponds to<br>`CALL CMSDT(CM_DEALLOCATE_CONFIRM) and CALL CMDEAL`<br><br>`CALL CMSERR`<br>**Note:** The SEND_ERROR call causes the Deallocate call to receive an error *return_code*. |
| The transaction ends if the remote program issues a<br>`RSPCONFIRM` | Corresponds to<br>`CALL CMCFMD` |
| If your program issues:<br>`DETACH with TNSSYNLVL`<br>If your program specified SYNLVL(*COMMIT), the transaction does not end until your program issues a:<br>`Commit operation`<br>to commit the changes. | Corresponds to:<br>`CALL CMSDT(CM_DEALLOCATE_SYNC_LEVEL) followed by CALL CMDEAL`<br><br>`Commit operation` |
| If a program receives the following detach and commit indications for a protected conversation<br>`RCVDETACH and RCVTKCMT`<br>(or it receives return codes 0259 or 0359), it should commit changes using:<br>`Commit operation` | Corresponds to:<br>`A status received of CM_TAKE_COMMIT_DEALLOCATE`<br><br>`Commit operation` |
| If the program is a target program, started by a program start request, and it receives a detach indication for an unprotected conversation using:<br>`RCVDETACH`<br>(or it receives return codes 0008, 0208, or 0308), it should end the session using:<br>`EOS or close` | Corresponds to<br>`A return code of CM_DEALLOCATED_NORMAL`<br><br>Not applicable |
| If your program uses an<br>`EOS`<br>during an active transaction, both the transaction and the conversation are ended abnormally by APPC. | Corresponds to<br>`CALL CMSDT(CM_DEALLOCATE_ABEND)`<br>`followed by CALL CMDEAL` |

## Flow Diagram for Inquiry Applications Using CPI Communications Calls

Figure 6-1 on page 6-13 shows the flow for the exchange of information that occurs in an inquiry application using CPI Communications calls. This flow parallels the one for ICF inquiry applications on page 5-17.

The steps shown in Figure 6-1 are described in Table 6-6 :

*Table 6-6 (Page 1 of 3). Description of Flow Diagram*

| Step | Description |
|---|---|
| 1 | 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:<br>• Initialize a conversation.<br>• Identify the name of the target program (using *sym_dest_name*).<br>• Ask CPI Communications to establish the identifier that the program uses when referring to the conversation (the *conversation_ID*).<br>On successful completion of the Initialize_Conversation call, CPI Communications provides the *conversation_ID* and returns it to Program A. The program must store the *conversation_ID* and use it on all subsequent calls intended for that conversation. |

*Table 6-6 (Page 2 of 3). Description of Flow Diagram*

| Step | Description |
|---|---|
| 2 | No errors occurred on the Initialize_Conversation call, so the *return_code* is set to CM_OK.<br>Two major tasks are now accomplished:<br>• CPI Communications has established a set of conversation characteristics for the conversation, based on the *sym_dest_name*, and uniquely associates them with the *conversation_ID*.<br>• Default values for the conversation characteristics are established. |
| 3 and 4 | Program A sets the *sync_level* characteristic to CM_CONFIRM by issuing the Set_Sync_Level call.<br>**Note:** Program A must set the *sync_level* characteristic before issuing the Allocate call (Step 9 ) if the new value is to take effect. Changing the *sync_level* after the conversation is allocated results in an error condition. |
| 5 and 6 | Program A sets the *send_type* characteristic to CM_SEND_AND_PREP_TO_RECEIVE by issuing the Set_Send_Type call. Setting the *send_type* to CM_SEND_AND_PREP_TO_RECEIVE means that Program A will be in Receive state after issuing a Send_Data call. |
| 7 and 8 | Program A sets the *deallocate_type* characteristic to CM_DEALLOCATE_FLUSH by issuing the Set_Deallocate_Type call.<br>**Note:** The default value for the *deallocate_type* conversation characteristic is CM_DEALLOCATE_SYNC_LEVEL. A Deallocate call issued with a *sync_level* of CM_CONFIRM would start the Confirm call and if successful, deallocate the conversation normally. |
| 9 and 10 | Program A issues the Allocate call by using the *conversation_ID* 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.<br>A *return_code* 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. |
| 11 | Program A sends data (the part number) with the Send_Data call.<br>**Note:** The Send_Data call is issued with the following conversation characteristics: 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. |
| 12 and 13 | Program C responds by issuing a call to Accept_Conversation, which places the conversation into Receive state. The Accept_Conversation call is similar to the Initialize_Conversation call because it equates a *conversation_ID* with a set of conversation characteristics. Program C, like Program A in Step 2 , receives a unique *conversation_ID* 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 *send_type* characteristic to CM_SEND_AND_PREP_TO_RECEIVE by issuing the Set_Send_Type call. |
| 16 and 17 | Program C sets the *prepare_to_receive_type* conversation characteristic to CM_PREP_TO_RECEIVE_FLUSH by issuing the Set_Prepare_To_Receive_Type call. |
| 18 | Program C issues the Receive call to receive the data. |
| 19 | A *return_code* of CM_OK indicates successful receipt of data. Program C also receives a request for confirmation, indicated by the *status_received* parameter, which is set to CM_CONFIRM_SEND_RECEIVED. 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 Confirmed call, which completes the program as send data with a return_code of CM_OK. |
| 22 | Program A issues a Receive call to receive the data. |
| 23 | Program C sends the data that was requested using the Send_Data call, and is now in Receive state because the *send_type* was set to CM_SEND_AND_PREP_TO_RECEIVE. The data and the permission-to-send indication are transmitted to Program A.<br>The Receive 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 Receive call to receive the data. |

*Table 6-6 (Page 3 of 3). Description of Flow Diagram*

| Step | Description |
|---|---|
| 25 | Program A issues a Deallocate call to release the conversation. Because the *deallocate_type* characteristic was set to CM_DEALLOCATE_FLUSH (see Step 7 ), the local data buffer of the LU is flushed and the conversation is deallocated normally.<br><br>The partner program (Program C) receives the deallocate notification by means of a *return_code* of CM_DEALLOCATED_NORMAL. |
| 26 | A return code of CM_OK tells Program A that the conversation is successfully deallocated. Both Program A and Program C complete normally. |

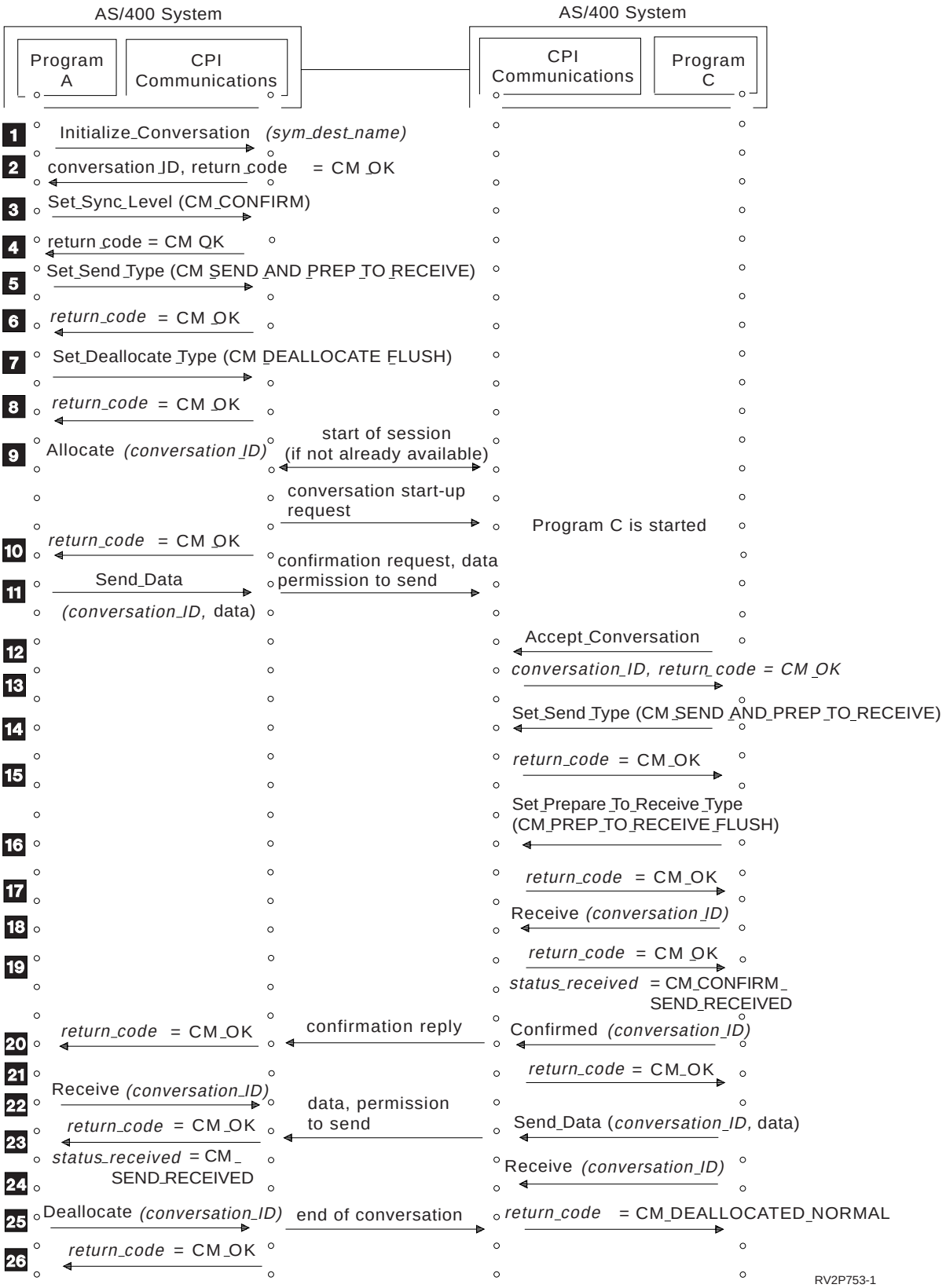



*Figure 6-1. Data Flow Using CPI Communications Calls*

## 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. Table 6-7 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.

*Table 6-7 (Page 1 of 2). 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 |
| BACKED_OUT | CM_TAKE_BACKOUT |
| BACKED_OUT ALL_AGREED | None |
| BACKED_OUT LUW_OUTCOME_MIXED | None |
| BACKED_OUT LUW_OUTCOME_PENDING | None |
| CONVERSATION_TYPE_MISMATCH | CM_CONVERSATION_TYPE_MISMATCH |
| DEALLOCATE_ABEND | CM_DEALLOCATED_ABEND |
| DEALLOCATE_ABEND BACKOUT_REQUIRED | CM_DEALLOCATED_ABEND_BO |
| DEALLOCATE_ABEND_PROG | CM_DEALLOCATED_ABEND_PROG |
| DEALLOCATE_ABEND_PROG BACKOUT_REQUIRED | CM_DEALLOCATED_ABEND_PROG_BO |
| DEALLOCATE_ABEND_SVC | CM_DEALLOCATED_ABEND_SVC |
| DEALLOCATE_ABEND_SVC BACKOUT_REQUIRED | CM_DEALLOCATED_ABEND_SVC_BO |
| DEALLOCATE_ABEND_TIMER | CM_DEALLOCATED_ABEND_TIMER |
| DEALLOCATE_ABEND_TIMER BACKOUT_REQUIRED | CM_DEALLOCATED_ABEND_TIMER_BO |
| 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_NO_RETRY BACKOUT_REQUIRED | CM_RESOURCE_FAIL_NO_RETRY_BO |
| RESOURCE_FAILURE_RETRY | CM_RESOURCE_FAILURE_RETRY |
| RESOURCE_FAILURE_RETRY BACKOUT_REQUIRED | CM_RESOURCE_FAILURE_RETRY_BO |
| 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 |

*Table 6-7 (Page 2 of 2). Return Code Mapping for CPI Communications and LU 6.2*

| LU 6.2 Return Code | CPI Communications Return Code |
|---|---|
| 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.
  - A minor return code indicating that a commit or rollback request has been received. The RCVTKCMT or RCVROLLB keyword may also be used.
- 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.
- The maximum length of a request unit (RU) is 32K bytes.

## 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, providing 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 toUsing 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 APPC support waits 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.

## WAITFILE Considerations for Switched Connections

Acquire operations and evoke functions rarely time out when a switched connection (such as an SDLC switched line, an X.25 SVC, an Ethernet connection, or a token-ring connection) is used without APPN support (APPN(*NO)). In this case (switched connection without APPN support), the system only has to find the device that matches the parameters of the ICF device entry before the WAITFILE timer runs out. Therefore, unless your system is very slow, the WAITFILE timer never runs out. However, if the device already has one or more active conversations, the WAITFILE timer also times the BIND command, which is sent to the remote system. In this case (active conversations), the acquire operation or evoke function may time out.

The WAITFILE value is not used when there are no active sessions for an acquired device using a switched connection, such as, an SDLC switched line, and 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, and one or more sessions are active, the WAITFILE parameter is used.

Acquire operations and evoke functions may or may not time out when a switched connection is used with APPN support (APPN(*YES)). In this case (switched connection with APPN support), the WAITFILE timer times the following system activities:

1. Finding the device that matches the parameters of the ICF device entry.
2. Verifying the remote location if the remote location name does not match the remote control point name. (A message is sent to the remote system, which must respond appropriately.)
3. Dialing the line for the first hop.
4. Creating a device if necessary.
5. Varying on the device if necessary.

After the system has successfully done the preceding five steps, it sends the BIND command. Normally, the BIND command also completes successfully. However, the WAITFILE timer does not time the BIND if a switched line is used and the device does not have any active conversations. (This is the same for switched connections with or without APPN support.) Therefore, if the system does not receive a response to the BIND command and the device does not have any active conversations, the acquire operation or evoke function does not time out and control does not return to the application.

When an evoke function follows a successful detach function that was already sent or received, the system sends a BIND command. The WAITFILE timer times this BIND command. If the switched line has already disconnected, the evoke function fails.

## WAITFILE Considerations for APPN Support

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 32K 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 32KB 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 5-5.

## Input Considerations

- The response 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 keep 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 VARBUFMGT 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.

## Two-Phase Commit Considerations

The following should be considered when programming for two-phase commit.

### Committing Resources

Your program requests that protected resources are committed by using the commit operation or the PRPCMT function.

Your program is notified that it has received a commit request from the remote program in the following ways:

- A major return code of 02 or 03 with minor return codes of 57, 58, or 59.
- The RCVTKCMT response indicator is set. The RCVTRNRND and RCVDETACH response indicators may also be set.

When your program receives a commit request, it must respond positively or negatively to the request as follows:

- To respond positively, do a commit operation.
- To respond negatively to the request, do one of the following:
  - Do a rollback operation.
  - Issue a fail function. This causes the logical unit of work to be rolled back if the partner issued a commit operation. Otherwise, if the partner issued a prepare-for-commit function, your program can attempt error recovery.
  - Abnormally end the transaction and session by issuing either an end-of-session function or a close operation.

### Rolling Back Resources

Your program requests that protected resources are rolled back by using the rollback operation.

Your program is notified that it has received a rollback request from the remote program or that rollback is required because of an error on the conversation in the following ways.

- One of the following return codes is received.
  - 0054
  - 0254
  - 80F9, 80FA, 80FB
  - 81F0, 81F1, 81F2, 81F3, 81F4, 81F5
  - 83FB, 83FC, 83FD, 83FE, 83FF
- The RCVROLLB response indicator is set.

When your program receives a rollback request, it must respond with a rollback operation.

### Exchanging Log Names

APPC uses a mechanism called exchange log name to negotiate the exact two-phase commit capabilities used for a protected conversation. The two systems at each end of the conversation exchange information about their level of two-phase commit support. Together, they decide which functions to use.

Exchange log name processing is performed when the system attempts to evoke its first protected conversation after communications have been established between the two systems. (The active session count between the two systems goes from 0 to 1.) The evoke is pended until the exchange log name processing has completed successfully.

Exchange log name processing brings up its own session to do the negotiation. You need to configure the mode description with one extra session that the system can use for exchange log name processing.

### Performance

The following are performance considerations for two-phase commit processing.

- The first protected conversation evoke between two systems takes longer to complete because of the exchange log name processing that takes place between the systems.
- The user may experience slower response times due to the two-phase commit processing needed to process the commit and rollback operations. Commit and rollback operations are done for each transaction program in the two-phase commit transaction program network.
- The bigger the two-phase commit transaction program network and the greater the number of commits issued for each transaction, the slower the response time.
- If data integrity is critical to your application, you should use two-phase commit processing. The extra processing that is done to ensure data integrity slows the performance of applications that use two-phase commit processing.

## 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.
- Protected conversations must commit the current logical unit of work before they can end normally. If an end-of-session function is issued before the logical unit of work is committed, APPC rolls back the logical unit of work.
- Application programs using protected conversations should detach the conversation before ending the session. The DETACH keyword issued with the TNSSYNLVL keyword and followed by a commit operation causes the system to commit all changes in the current logical unit of work and end the conversations. Ignoring this advice and issuing a write with EOS when there are still active protected conversations causes APPC to do the following.
  - Issue a DEALLOCATE_ABEND on those conversations.
  - Roll back the current logical unit of work.

## 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 Programming* book. For information on the prestart job entry format, refer to the *Work Management* book.

Prestart jobs can be used for protected conversations. However, if a prestart job attempts an acquire operation to the requesting program (*REQUESTOR) device and the prestart job has already evoked protected conversations, the acquire operation is rejected. The prestart job must end any protected conversations before it can acquire the *REQUESTOR device. This only applies when protected conversations are already active. A prestart job can acquire the *REQUESTOR device if there are unprotected conversations already active.

## 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 Status (DSPSRVSTS) commands.

For more information about the Trace ICF command, refer to the *ICF Programming* book.

# 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.
- The maximum length of a request unit (RU) is 32K bytes.

## Performance Considerations

- APPC can buffer up to 32KB 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” on page 8-2 for information on reducing this time delay.
- For slow lines, data compression may improve performance. For more information, see “APPC Data Compression” on page 3-6.

## Two-Phase Commit Considerations

The following should be considered when programming for two-phase commit.

## Committing Resources

Your program requests that protected resources are committed by using the commit operation.

Your program is notified that it has received a commit request from the remote program by the following return codes.

- CM_TAKE_COMMIT
- CM_TAKE_COMMIT_SEND
- CM_TAKE_COMMIT_DEALLOCATE

When your program receives a commit request, it must respond positively or negatively to the request as follows:

- To respond positively, do a commit operation.
- To respond negatively to the request, do one of the following:
  - Do a rollback operation.
  - Call CMSERR. In this case, your program is responsible for initiating the appropriate level of error recovery.
  - Call CMSDT with CM_DEALLOCATE_ABEND. Then call CMDEAL.

## Rolling Back Resources

Your program requests that protected resources are rolled back by using the rollback operation.

Your program is notified that it has received a rollback request from the remote program or that rollback is required because of an error on the conversation by the following return codes.

- CM_DEALLOCATED_ABEND_BO
- CM_DEALLOCATED_ABEND_PROG_BO
- CM_DEALLOCATED_ABEND_SVC_BO
- CM_DEALLOCATED_ABEND_TIMER_BO
- CM_RESOURCE_FAIL_NO_RETRY_BO
- CM_RESOURCE_FAILURE_RETRY_BO
- CM_TAKE_BACKOUT

When your program receives a rollback request, it must respond with a rollback operation.

## Exchanging Log Names

APPC uses a mechanism called exchange log name to negotiate the exact two-phase commit capabilities used for a protected conversation. The two logical units at each end of the conversation exchange information about their level of two-phase commit support. Together, they decide which functions to use.

Exchange log name processing is performed when the system attempts to allocate its first protected conversation after communications have been established between the two systems. (The active session count between the two systems goes from 0 to 1.) The allocate is pended until the exchange log name processing has completed successfully.

Exchange log name processing brings up its own session to do the negotiation. You need to configure the mode description with one extra session that the system can use for exchange log name processing.

## Immediate Return on Allocate

Because two-phase commit processing requires the exchange log name processing to complete before allocating protected conversations, conversations allocated with a *return_control* of CM_IMMEDIATE do not return control to the calling program immediately.

## Performance

The following are performance considerations for two-phase commit processing.

- Allocation of the first protected conversation between two systems takes longer to complete because of the exchange log name processing that takes place between the systems.
- The user may experience slower response times due to the two-phase commit processing needed to process the commit and rollback operations. Commit and rollback operations are done for each transaction program in the two-phase commit transaction program network.
- The bigger the two-phase commit transaction program network and the greater the number of commits issued for each transaction, the slower the response time.
- If data integrity is critical to your application, you should use two-phase commit processing. The extra processing that is done to ensure data integrity slows the performance of applications that use two-phase commit processing.

## 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.
- Prestart jobs can be used for protected conversations. However, if a prestart job attempts an Accept_Conversation (CMACCP) call and the prestart job has already allocated protected conversations, the CMACCP call is rejected. The prestart job must end any protected conversations before it can accept an incoming protected conversation. This only applies when protected conversations are already active. A prestart job can accept an incoming protected conversation if there are unprotected conversations already active.

See the *Work Management* book 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 Status (DSPSRVSTS) commands.

For more information about the Trace CPI Communications command, refer to the *Communications Management* book.

# Appendix A. ICF Operations, DDS 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

Table A-1 describes the language operations supported by ICF.

*Table 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.)

Table A-2 on page A-2 presents the ICF file operations used with APPC communications and the equivalent high-level language statement.

*Table A-2. High-Level Language I/O Operations*

| ICF Operation | ILE RPG/400 Operation Code | ILE COBOL/400 Procedure Statement | ILE C/400 Function[1] | FORTRAN/400 Statement |
|---|---|---|---|---|
| Open | OPEN | OPEN | fopen or _Ropen | OPEN |
| Acquire | ACQ | ACQUIRE | _Racquire | Not supported[2] |
| Get attributes | POST | ACCEPT | _Rdevatr | Not supported |
| Read | READ | READ | fread or _Rreadn | READ |
| Read-from-invited-program-devices | READ[3] | READ[3] | _Rreadindv | Not supported |
| Write | WRITE | WRITE | fwrite or _Rwrite | WRITE |
| Write/Read | EXFMT | Not supported | _Rwriterd | Not supported |
| Release | REL | DROP | _Rrelease | Not supported |
| Close | CLOSE | CLOSE | fclose or _Rclose | CLOSE |

**Note:**

1 ILE 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 program device 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

Table A-3 lists the DDS keywords that are valid for APPC.

*Table 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. |
| DFREVOKE | Delays an evoke until one of the following conditions is met.<br>• The send buffer is full.<br>• An ALWWRT, CONFIRM, DETACH, FRCDTA, INVITE, PRPCMT, or READ function is issued.<br>• A commit or rollback operation is done. |
| 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. |

*Table A-3 (Page 1 of 2). DDS Keywords*

| DDS Keyword | Description |
|---|---|
| INVITE | Schedules an invite function. |
| PRPCMT | Requests that the remote program prepare to commit its protected resources. |
| 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. |
| RCVROLLB | Indicates that the local program needs to rollback its protected resources. |
| RCVTKCMT | Indicates that the local program needs to determine if it can commit its protected resources. |
| 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. |

*Table A-3 (Page 2 of 2). DDS Keywords*

| DDS Keyword | Description |
|---|---|
| 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. |
| TNSSYNLVL | Specifies that synchronization for this transaction should be done at the level that the SYNLVL keyword specified on the evoke. |
| VARBUFMGT | 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

Table A-4 lists all the keyword functions performed by the system-supplied formats that are valid for APPC. Refer to the *ICF Programming* book for more information about system-supplied formats.

*Table 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

## 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 Support* book 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 *partnerprogram* 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 turnaroundindication. 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 Table B-1:

*Table B-1. 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. |

**Note:**

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:** For a receive operation, 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 turnaroundindication (which allows your program to send data) or a detach indication.

For an acquire operation, your program received control data, sent with the program start request.

**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 partnerprogram 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 turnaroundindication.

**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 partnerprogram 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 turnaroundindication. 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.

**0054** **Description:** Rollback is required. The transaction program (TP) has entered the rollback-required state.

**Note:** This is only returned after an EOS function or a close operation has been issued, so that local resources and other remote resources must be rolled back.

**Action:** Use the rollback operation to roll back all local protected resources.

## 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** | **Description:** On a successful input operation, your program received some data with a turnaround indication. Also, your job is being ended (controlled) The partnerprogram is ready to receive data from your program.<br><br>**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. |
| **0201** | **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.<br><br>**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. |
| **0202** | **Description:** For a receive operation, 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 turnaroundindication (which allows your program to send data) or a detach indication.<br><br>For an acquire operation, your program received control data, sent with the program start request.<br><br>**Action:** Your program can issue another input operation. If your program detects the equivalent of a turnaroundindication, 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. |
| **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 partnerprogram is ready to receive data from your program.<br><br>**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.<br><br>**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 turnaroundindication. Also, your job is being ended (controlled) The partner program is ready to receive data from your program.<br><br>**Action:** Your program can issue an outputoperation. 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 partnersystem is still active. Also, your job is being ended (controlled)<br><br>**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 turnaroundindication. 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 turn-around indication. In addition, the partnerprogram 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 outputoperation. 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 partnerprogram

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 partnerprogram 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.

**0254** **Description:** Rollback is required. The transaction program (TP) has entered the rollback-required state.

**Note:** This is only returned after an EOS function or a close operation has been issued, so that local resources and other remote resources must be rolled back.

Also, your job is being ended (controlled)

**Action:** Use the rollback operation to roll back all local protected resources.

**0257** **Description:** The remote program has issued either a commit operation or a prepare-for-commit function. This requests the local program to respond by issuing a commit operation in order to perform the two-phase commit processing on all protected resources. Also, your job is being ended (controlled)

**Action:** Do a commit operation to cause a positive response to be returned to the remote program if the two-phase commit processing was successful. To respond negatively, do a rollback operation, an EOS function, or a fail function.

**0258** **Description:** The remote program has issued an allow-write function with the transaction-synchronization-level function followed by either a commit operation or a prepare-for-commit function. The synchronization level is *COMMIT. Your program will be in send state after issuing a commit operation, once the commit operation completes successfully. Also, your job is being ended (controlled)

**Action:** Do a commit operation to cause a positive response to be returned to the remote program if the two-phase commit processing was successful. To respond negatively, do a rollback operation, an EOS function, or a fail function.

**0259** **Description:** The remote program has issued a detach function with the transaction-synchronization-level function followed by either a commit operation or a prepare-for-commit function. The synchronization level is *COMMIT. Your program will be deallocated after issuing a commit operation, once the commit operation completes successfully. Also, your job is being ended (controlled)

**Action:** Do a commit operation to cause a positive response to be returned to the remote program if the two-phase commit processing was successful. To respond negatively, do a rollback operation, an EOS function, or a fail function.

## 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** **Description:** On a successful input operation, your program received a turnaround indication without any data. The session is still active.

**Action:** Issue an output operation.

**0301** **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.

**Action:** Issue an input operation.

**0302** **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.

**Action:** Issue an input operation.

**0304** **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.

**Action:** Issue an output operation.

**0305** **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.

**Action:** Issue an input operation.

**0306** **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.

**Action:** Issue an output operation.

**0308** **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 partnersystem 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.

**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 partnersystem 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 partnerprogram 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 partnerprogram 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 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.

**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.

**0357** **Description:** The remote program has issued either a commit operation or a prepare-for-commit function. This requests the local program to respond by issuing a commit operation in order to perform the two-phase commit processing on all protected resources.

**Action:** Do a commit operation to cause a positive response to be returned to the remote program if the two-phase commit processing was successful. To respond negatively, do a rollback operation, an EOS function, or a fail function.

**0358** **Description:** The remote program has issued an allow-write function with the transaction-synchronization-level function followed by either a commit operation or a prepare-for-commit function. The synchronization level is *COMMIT. Your program will be in send state after issuing a commit operation, once the commit operation completes successfully.

**Action:** Do a commit operation to cause a positive response to be returned to the remote program if the two-phase commit processing was successful. To respond negatively, do a rollback operation, an EOS function, or a fail function.

**0359** **Description:** The remote program has issued a detach function with the transaction-synchronization-level function followed by either a commit operation or a prepare-for-commit function. The synchronization level is *COMMIT. Your program will be deallocated after issuing a

commit operation, once the commit operation completes successfully.

**Action:** Do a commit operation to cause a positive response to be returned to the remote program if the two-phase commit processing was successful. To respond negatively, do a rollback operation, an EOS function, or a fail function.

## 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** **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.

**Action:** Correct the error that caused your program to select a keyword that was not valid.

**Messages:**

- CPF5064 (Notify)

**0412** **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.

**Action:** Issue an input operation to receive the data.

**Messages:**

- CPF4750 (Notify)
- CPF5076 (Notify)
- CPF83D6 (Escape)

## 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** **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.

**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.

**Messages:**

- CPD4077 (Diagnostic)
- CPF5041 (Status)
- CPF50A0 (Status)

**1100** **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.

**Action:** Issue an invite function (or a combined operation that includes an invite) followed by a read-from-invited-program-devices operation.

**Messages:**

- 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** **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.

**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 keep overflow data.

**Messages:**

- CPF4936 (Notify)

**3422** **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.

**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 keep overflow data.

**Messages:**

- CPF4936 (Notify)

**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 keep 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 kept 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.

**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 kept 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.

## 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** **Description:** The operation attempted by your program was not successful because a system error condition was detected.

**Action:** Your communications configurations may need to be varied off and then on again. Your program can do one of the following:

- Continue local processing.
- Close the ICF file, open the file again, and establish the session again.
- End.

**Messages:**

- CPF4061 (Diagnostic)
- CPF4170 (Escape)
- CPF4323 (Escape)
- CPF4510 (Escape)
- CPF5257 (Escape)
- CPF5411 (Escape)
- CPF5424 (Escape)
- CPF5425 (Escape)
- CPF5449 (Escape)
- CPF5537 (Escape)
- CPF5E7D (Escape)
- CPF5E7E (Escape)
- CPF5E7F (Escape)
- CPF8350 (Escape)
- CPF8362 (Escape)
- CPF8364 (Escape)
- CPF8365 (Escape)
- CPF8367 (Escape)

**8082** **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.

**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 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 fileas 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)

**80F9** **Description:** The operation attempted by your program was not successful because a system error condition was detected. Rollback required.

**Action:** Your program must do a rollback operation. Your communications configurations may need to be varied off and then on again. Your program can do one of the following:

- Continue local processing.
- Close the ICF file, open the file again, and establish the session again.
- End.

**Messages:**

- CPF4061 (Diagnostic)
- CPF4170 (Escape)
- CPF4323 (Escape)
- CPF4510 (Escape)
- CPF5257 (Escape)
- CPF5411 (Escape)
- CPF5424 (Escape)
- CPF5425 (Escape)
- CPF5449 (Escape)
- CPF5537 (Escape)

**80FA** **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. Rollback required.

**Action:** Your program must do a rollback operation. 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 and then on again. Your program can attempt to establish the session again, continue local processing, or end.

**Messages:**

- CPF4744 (Escape)
- CPF5269 (Escape)

**80FB** **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 81F3 and 81F4 are translated into 80FB.

Rollback required.

**Action:** Your program must do a rollback operation. 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)

## Major Code 81

**Major Code 81** – Permanent session error (irrecoverable).

**Description:** An irrecoverable 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 started 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 started by a program start request from the partnerprogram, 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**

**8101** **Description:** The protected password could not be built for the evoke operation. The password for the signed-on user was retrieved from the security file. Any of the following conditions can prevent the protected password from being built.

- The password was last changed on a release earlier than Version 2 Release 2.
- The user profile does not exist on the system.
- The password is *NONE.
- The user profile name is not valid.
- The user profile is disabled.

**Action:** Make sure the password for the signed-on user is:

- A value other than *NONE.
- Has been changed on a system running OS/400 Version 2 Release 2 or later.

Make sure the user profile for the signed-on user is:

- On the system.
- Named properly.
- Not disabled.

Run your program again.

**Messages:**

- CPF512A (Escape)

**8140** **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.

**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.

**Messages:**

- CPF5104 (Escape)

**8191** **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.

**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.

**Messages:**

- CPF4146 (Escape)
- CPF410E (Escape)
- CPF4155 (Escape)
- CPF4193 (Escape)
- CPF4291 (Escape)
- CPF510E (Escape)
- CPF5128 (Escape)
- CPF5198 (Escape)
- CPF5544 (Escape)

**8196** **Description:** The partner system sent a Systems Network Architecture (SNA) UNBIND command to your system, or the session was ended locally.

**Action:** To start another session, issue the acquire operation again. Otherwise, your program can continue local processing or end.

**Messages:**

- CPF4060 (Diagnostic)
- CPF4145 (Escape)
- CPF4299 (Escape)
- CPF5107 (Escape)
- CPF5166 (Escape)
- CPF5524 (Escape)

**8197** **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.

**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.

**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 partnersystem 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 partnersystem abnormally ended the session (TYPE=TIMER). For example, the partnerprogram 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)

**81F0** **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 message queue (QSYSOPR). The session has ended. Data may have been lost. Rollback required.

**Action:** Do a rollback operation. 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.

**Messages:**

- CPF410E (Escape)
- CPF4146 (Escape)
- CPF4155 (Escape)
- CPF4193 (Escape)
- CPF4291 (Escape)
- CPF510E (Escape)
- CPF5128 (Escape)
- CPF5198 (Escape)
- CPF5544 (Escape)

**81F1** **Description:** The partner system sent a Systems Network Architecture (SNA) UNBIND command to your system, or the session was ended locally. Rollback required.

**Action:** Do a rollback operation. To start another session, issue the acquire operation again. Otherwise, your program can continue local processing or end.

**Messages:**

- CPF4060 (Diagnostic)
- CPF4145 (Escape)
- CPF4299 (Escape)
- CPF5107 (Escape)
- CPF5166 (Escape)
- CPF5524 (Escape)

**81F2** **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. Rollback required.

**Action:** Do a rollback operation. 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.

**Messages:**

- CPF4145 (Escape)
- CPF5107 (Escape)
- CPF5283 (Escape)
- CPF5526 (Escape)

**81F3** **Description:** The partner program or partnersystem abnormally ended the session (TYPE=SVC). Rollback required.

**Action:** Do a rollback operation. Contact the remote system to determine why the error occurred.

**Messages:**

- CPF4145 (Escape)
- CPF5107 (Escape)
- CPF5283 (Escape)

**81F4** **Description:** The partner program or partnersystem abnormally ended the session (TYPE=TIMER). For example, the partnerprogram may have been canceled by the operator. Rollback required.

**Action:** Do a rollback operation. Contact the remote system to determine why the error occurred.

**Messages:**

- CPF4145 (Escape)
- CPF5107 (Escape)
- CPF5283 (Escape)

**81F5** **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. Rollback required.

**Action:** Do a rollback operation. Verify that the data sent by the partner program was correct. If the data was not correct, have the operator on the partnersystem 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 started by a program start request from the partnerprogram, 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** **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:

- 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.

**Action:** Complete all processing and end your program as soon as possible. Correct the system error before starting this job again.

**Messages:**

- CPF4292 (Escape)
- CPF5313 (Escape)

**8233** **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 commandfor the session being acquired. The session was not started.

**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.

**Messages:**

- CPF4288 (Escape)
- CPF5068 (Escape)

**8281** **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.

**Action:** Your communications configurations may need to be varied off and then on again. Your program can do one of the following:

- 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.

**Messages:**

- CPF4106 (Escape)
- CPF4168 (Escape)
- CPF4182 (Escape)
- CPF4369 (Escape)
- 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 dedicated service tools (DST) are 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 commandwas 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)

**82FA** **Description:** The local LU rejected the allocation request because the local program specified a synchronization level (on the evoke function) that the remote LU does not support.

**Action:** Try the evoke function again using a synchronization level that is supported by the remote system.

**Messages:**

- CPF510A (Escape)

**82FB** **Description:** Protected conversations are not supported on single-session devices.

**Action:** Either change the single session (SNGSSN) parameter of the device description

to *NO, or use a different device description that specifies SNGSSN(*NO).

**Messages:**

- CPF501C (Notify)

**82FC** **Description:** Protected conversations are not supported by the System/36 and System/38 environments.

**Action:** Do not try to use protected conversations while using the System/36 or System/38 environments.

**Messages:**

- CPF413D (Escape)
- CPF413E (Escape)
- CPF512B (Escape)
- CPF512C (Escape)

**82FD** **Description:** The exchange log name process failed.

**Action:** Make sure the log names specified by the local and remote system can be used together. Then run your program again.

**Messages:**

- CPF5E13 (Notify)
- CPF5E15 (Escape)

**82FE** **Description:** The evoke function issued by your program was not successful because a resource could not be placed under commitment control.

**Action:** Refer to the recovery text of message CPF8361.

**Messages:**

- CPF8361 (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 partnersystem 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** **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.

**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.

**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.
- Your program issued a detach function for a protected conversation without using the transaction-synchronization-level function.

**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:**

- CPF470C (Notify)
- CPF470D (Notify)
- CPF470E (Notify)
- CPF470F (Notify)
- CPF4764 (Notify)
- CPF4766 (Notify)
- CPF4790 (Notify)
- CPF501D (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 indicate 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 partnerprogram 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 partnerprogram 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 partnerprogram 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 partnerprogram 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 partnerlocation to determine if the PIP data is defined correctly for the specified partnerprogram.

**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 partnerprogram 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 incorrect length specification on a basic conversation.

**Action:** Change your program or the record format to issue a valid 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)

**83FB** **Description:** Your program closed the ICF file while the transaction was still active. The system abnormally ended the transaction with the partner program. Rollback required.

**Action:** Do a rollback operation. 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)

**83FC** **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. Rollback required.

**Action:** Do a rollback operation. Release the program device, correct the previous error, then acquire the program device again.

**Messages:**

- CPF5293 (Escape)

**83FD** **Description:** All protected resources have rolled back in the part of the distributed transaction affected by the function.

**Action:** Use the rollback operation to roll back all local protected resources.

**83FE** **Description:** The state of one or more protected resources is not known. The changes probably are or will be rolled back, but changes to some resources may be committed instead. Your protected LUW is in the rollback required state.

**Action:** Use the rollback operation to roll back the protected LUW. To determine the outcome of the protected LUW, use the Display Commitment Control Status option of the Work with Job (WRKJOB) or Display Job (DSPJOB) commands.

**83FF** **Description:** The state of the protected resources is not consistent. One or more resource have advanced to a new synchronization point (have been committed instead of rolled back). Your protected LUW is in the rollback required state.

This return code occurs only when processing has been abnormally interrupted through operator intervention.

**Action:** Use the rollback operation to roll back the protected LUW. Make sure the protected LUW is in consistent states on all systems in the transaction program network. To display the status of the protected LUW, use the Display Commitment Control Status option of the Work with Job (WRKJOB) or Display Job (DSPJOB) commands. You may have to do this on each remote system as well.

## CPI Communications Return Codes

Refer to the *CPI Communications Reference* for information about CPI Communications return codes.

## Program Start Request Errors

Sense codes returned by failed program start requests (such as 084C0000, 084B6031, or 080F6051) indicate that the remote system has rejected a request to do work on that system.

When a program start request fails, the remote system will post a CPF1269 message in the QSYSMSG message queue

and the history log. If the QSYSMSG message queue does not exist, the remote system will post the CPF1269 message in the QSYSOPR message queue and the history log.

CPF1269 contains a reason code that describes why the program start request failed. For AS/400 systems running V2R3 or later, the meaning of the reason code is included in the message.

The SNA FMH7 sense data shown in Table B-2 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).

*Table B-2 (Page 1 of 3). Reason Codes for Rejected Program Start Requests*

| SNA FMH7 Sense Data | 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. |
| 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. |

*Table B-2 (Page 2 of 3). Reason Codes for Rejected Program Start Requests*

| SNA FMH7 Sense Data | Reason Code | Reason Description |
|---|---|---|
| 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. |
| 080F6051 | 727 | Protected password invalid; either the user ID or password was longer than 8 characters, or the password was created on a release prior to V2R3. |
| 084B0000 | 728 | No OS/400 license available. |
| 084B6031 | 729 | Program start request received for SYNLVL(*COMMIT) conversation before exchange log names completed. |
| 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. |
| 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. |

*Table B-2 (Page 3 of 3). Reason Codes for Rejected Program Start Requests*

| SNA FMH7 Sense Data | Reason Code | Reason Description |
|---|---|---|
| 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. |
| 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. |
| 10086051 | 2023 | The length of the attach sequence number is incorrect. It must be either 0 or 8 characters long. |
| 10086055 | 2024 | The attach sequence number is required, but is not present in the FMH5. |

# 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* book.

## 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 4-3.

**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 4-5.

**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:

- ILE COBOL/400: The identifier parameter of the ACQUIRE statement or the TERMINAL option of the WRITE and READ statements.
- ILE RPG/400: The program device identifier specified in factor 1 of the calculation specifications.
- ILE 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.

- SYNLVL DDS keyword (*NONE, *CONFIRM, or *COMMIT 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/SYNCPT)**

- SYNLVL DDS keyword (*NONE, *CONFIRM, or *COMMIT supported) specified with EVOKE keyword.
- For system-supplied format, support is always defaulted to NONE. System-supplied formats do not support CONFIRM or SYNCPT 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 3-12.

**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 the program is a source program and the conversation has a synchronization level of *COMMIT, a commit operation must follow the detach function and precede the end-of-session function, release operation, or close operation.

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)**
Use if conversation was allocated with SYNLVL (*COMMIT). It can also be used in place of TYPE(CONFIRM) for SYNLVL (*CONFIRM) conversations or in place of TYPE(FLUSH) for SYNLVL (*NONE) conversations.

- For data description specifications (DDS), use the TNSSYNLVL keyword.
- Not supported for system-supplied formats.

**TYPE(CONFIRM)**
Use if conversation was allocated with SYNLVL (*CONFIRM).

- For DDS, use the CONFIRM or TNSSYNLVL keyword.
- Not supported for system-supplied formats.

**TYPE(FLUSH)**
Use if conversation was allocated with SYNLVL (*NONE).

- For DDS, use the FRCDTA or TNSSYNLVL 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-attributes operation of ICF. Refer to “GET_ATTRIBUTES” on page C-11 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_FOR_SYNCPT:** The MC_PREPARE_FOR_SYNCPT verb causes a single protected resource to be prepared to advance to the next synchronization point. Using this verb allows a program to have its own logic to respond if its partner cannot complete the commit process. Support for MC_PREPARE_FOR_SYNCPT is provided by the PRPCMT DDS keyword. MC_PREPARE_FOR_SYNCPT 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 I/O feedback area

**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)**
Use if conversation was allocated with SYNLVL(*COMMIT). It can also be used in place of TYPE(CONFIRM) for SYNLVL(*CONFIRM) conversations or in place of TYPE(FLUSH) for SYNLVL(*NONE) conversations.

- For data description specifications (DDS), use the TNSSYNLVL keyword. The TNSSYNLVL keyword is not allowed with an invite function if the conversation was allocated with SYNLVL(*COMMIT).
- Not supported for system-supplied formats.

For SYNLVL(*COMMIT), the conversation enters defer receive state. Defer receive state indicates that this program wants to be in receive state after a commit operation completes successfully. The only valid operations for the conversation in defer receive state are: commit, rollback, PRPCMT, and EOS.

**TYPE(CONFIRM)**
Use if conversation was allocated with SYNLVL(*CONFIRM).

- For DDS, use the CONFIRM or TNSSYNLVL keyword.
- Not supported for system-supplied formats.

**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 confirmation. 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
- TAKE_COMMIT
- TAKE_COMMIT_SEND
- TAKE_COMMIT_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.

The following lists 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.

***ICF Return Codes Mapped to WHAT_RECEIVED Values:*** The following list shows ICF return codes with the corresponding WHAT_RECEIVED values for the MC_RECEIVE_AND_WAIT verb.

**ICF Return Code WHAT_RECEIVED Values**

| | |
|---|---|
| **0000** | DATA_COMPLETE, SEND |
| **0001** | DATA_COMPLETE |
| **0002** | CONTROL_DATA_COMPLETE |
| **0004** | CONTROL_DATA_COMPLETE, SEND |
| **0005** | CONTROL_DATA_COMPLETE |
| **0006** | CONTROL_DATA_COMPLETE, SEND |
| **0008**[1] | DATA_COMPLETE |
| **000C**[1] | CONTROL_DATA_COMPLETE |
| **0010** | DATA_COMPLETE |
| **0011**[1] | CONTROL_DATA_COMPLETE |
| **0013** | CONTROL_DATA_COMPLETE, CONFIRM_SEND |
| **0014** | DATA_COMPLETE, CONFIRM_SEND |
| **0015** | DATA_COMPLETE, CONFIRM |
| **0018** | CONTROL_DATA_COMPLETE, CONFIRM |
| **001C** | DATA_COMPLETE, CONFIRM_DEALLOCATE |
| **001D** | CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE |
| **0044** | CONTROL_DATA_COMPLETE, CONFIRM_SEND |

**0045** CONTROL_DATA_COMPLETE, CONFIRM

**0046** CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE

**0200** DATA_COMPLETE, SEND

**0201** DATA_COMPLETE

**0202** CONTROL_DATA_COMPLETE

**0204** CONTROL_DATA_COMPLETE, SEND

**0205** CONTROL_DATA_COMPLETE

**0206** CONTROL_DATA_COMPLETE, SEND

**0208**[1] DATA_COMPLETE

**020C**[1] CONTROL_DATA_COMPLETE

**0211**[1] CONTROL_DATA_COMPLETE

**0213** CONTROL_DATA_COMPLETE, CONFIRM_SEND

**0214** DATA_COMPLETE, CONFIRM_SEND

**0215** DATA_COMPLETE, CONFIRM

**0218** CONTROL_DATA_COMPLETE, CONFIRM

**021C** DATA_COMPLETE, CONFIRM_DEALLOCATE

**021D** CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE

**0244** CONTROL_DATA_COMPLETE, CONFIRM_SEND

**0245** CONTROL_DATA_COMPLETE, CONFIRM

**0246** CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE

**0257** TAKE_COMMIT

**0258** TAKE_COMMIT_SEND

**0259** TAKE_COMMIT_DEALLOCATE

**0300** SEND

**0301**

**0302** CONTROL_DATA_COMPLETE

**0304** CONTROL_DATA_COMPLETE, SEND

**0305** CONTROL_DATA_COMPLETE

**0306** CONTROL_DATA_COMPLETE, SEND

**0308**[1]

**030C**[1] CONTROL_DATA_COMPLETE

**0311**[1] CONTROL_DATA_COMPLETE

**0313** CONTROL_DATA_COMPLETE, CONFIRM_SEND

**0314** CONFIRM_SEND

**0315** CONFIRM

**0318** CONTROL_DATA_COMPLETE, CONFIRM

**031C** CONFIRM_DEALLOCATE

**031D** CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE

**0344** CONTROL_DATA_COMPLETE, CONFIRM_SEND

**0345** CONTROL_DATA_COMPLETE, CONFIRM

**0346** CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE

**0357** TAKE_COMMIT

**0358** TAKE_COMMIT_SEND

**0359** TAKE_COMMIT_DEALLOCATE

**3421** CONTROL_DATA_TRUNCATED

**3422** CONTROL_DATA_TRUNCATED

**3431** DATA_TRUNCATED

**3471** DATA_INCOMPLETE

**3481** CONTROL_DATA_INCOMPLETE

[1]The LU 6.2 architected return code DEALLOCATE is returned by this ICF return code.

***WHAT_RECEIVED Values Mapped to ICF Return Codes:*** The following list shows the WHAT_RECEIVED values for the MC_RECEIVE_AND_WAIT verb with the corresponding ICF return codes.

**WHAT_RECEIVED Value ICF Return Codes**

**DATA_COMPLETE** 0000, 0001, 0008[1], 0010, 0014, 0015, 001C, 0200, 0201, 0208[1], 0214, 0215, 021C

**DATA_TRUNCATED** 3431

**DATA_INCOMPLETE** 3471

**CONTROL_DATA_COMPLETE** 0002, 0004, 0005, 0006, 000C[1], 0011[1], 0013, 0018, 001D, 0044, 0045, 0046, 0202, 0204, 0205, 0206, 020C[1], 0211[1], 0213, 0218, 021D, 0244, 0245, 0246, 0302, 0304, 0305, 0306, 030C[1], 0311[1], 0313, 0318, 031D, 0344, 0345, 0346

**CONTROL_DATA_TRUNCATED** 3421, 3422

**CONTROL_DATA_INCOMPLETE** 3481

**SEND** 0000, 0004, 0006, 0200, 0204, 0206, 0300, 0304, 0306

**CONFIRM** 0015, 0018, 0045, 0215, 0218, 0245, 0315, 0318, 0345

**CONFIRM_SEND** 0013, 0014, 0044, 0213, 0214, 0244, 0313, 0314 0344

**CONFIRM_DEALLOCATE** 001C, 001D, 0046, 021C, 021D, 0246, 031C, 031D, 0346

**TAKE_COMMIT** 0257, 0357

**TAKE_COMMIT_SEND** 0258, 0358

**TAKE_COMMIT_DEALLOCATE** 0259, 0359

[1]The LU 6.2 architected return code DEALLOCATE is returned by this ICF return code.

***Response Indicators Mapped to WHAT_RECEIVED Values:*** The following list shows the DDS response indicators with the corresponding WHAT_RECEIVED values for the MC_RECEIVE_AND_WAIT verb.

**Response Indicators WHAT_RECEIVED Values**

**RCVCONFIRM** CONFIRM, CONFIRM_SEND, CONFIRM_DEALLOCATE

**RCVTRNRND** SEND, CONFIRM_SEND, TAKE_COMMIT_SEND

**RCVDETACH** CONFIRM_DEALLOCATE, TAKE_COMMIT_DEALLOCATE

**RCVCTLDTA** CONTROL_DATA_COMPLETE, CONTROL_DATA_TRUNCATED, CONTROL_DATA_INCOMPLETE

**RCVTKCMT** TAKE_COMMIT, TAKE_COMMIT_SEND, TAKE_COMMIT_DEALLOCATE

***Data Received with the Program Start Request:*** The following list shows ICF return codes with the corresponding WHAT_RECEIVED values for the MC_RECEIVE_AND_WAIT verb. These ICF return codes also indicate that data was received with the program start request.

**ICF Return Code WHAT_RECEIVED Values**

**0002** CONTROL_DATA_COMPLETE

**0006** CONTROL_DATA_COMPLETE, SEND

**0011**[1] CONTROL_DATA_COMPLETE

**0013** CONTROL_DATA_COMPLETE, CONFIRM_SEND

**0018** CONTROL_DATA_COMPLETE, CONFIRM

**001D** CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE

**0202** CONTROL_DATA_COMPLETE

**0206** CONTROL_DATA_COMPLETE, SEND

**0211**[1] CONTROL_DATA_COMPLETE

**0213** CONTROL_DATA_COMPLETE, CONFIRM_SEND

**0218** CONTROL_DATA_COMPLETE, CONFIRM

**021D** CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE

**0302** CONTROL_DATA_COMPLETE

**0306** CONTROL_DATA_COMPLETE, SEND

**0311**[1] CONTROL_DATA_COMPLETE

**0313** CONTROL_DATA_COMPLETE, CONFIRM_SEND

**0318** CONTROL_DATA_COMPLETE, CONFIRM

**031D** CONTROL_DATA_COMPLETE, CONFIRM_DEALLOCATE

**3422** CONTROL_DATA_TRUNCATED

[1]The 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.

If the conversation has a synchronization level of commit, the fail function may cause the system to roll back the protected LUW. If the fail function is in response to a commit operation, the system rolls back the protected LUW. If the fail function is in response to a prepare-for-commit function, the system does *not* roll back the protected LUW.

**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 Table C-1 on page C-11 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 param-

eter 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/SYNCPT)**

- SYNLVL DDS keyword (*NONE, *CONFIRM, or *COMMIT supported) specified with EVOKE keyword.
- For system-supplied format, support is always defaulted to NONE. System-supplied formats do not support CONFIRM or SYNCPT 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 3-12.

**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 the program is a source program and the conversation has a synchronization level of *COMMIT, a commit operation must follow the detach function and precede the end-of-session function, release operation, or close operation.

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)**
Use if conversation was allocated with SYNLVL (*COMMIT). It can also be used in place of TYPE(CONFIRM) for SYNLVL (*CONFIRM) conversations or in place of TYPE(FLUSH) for SYNLVL (*NONE) conversations.

- For data description specifications (DDS), use the TNSSYNLVL keyword.
- Not supported for system-supplied formats.

**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.

**TYPE(FLUSH)**
This is done implicitly for ICF. Use if conversation was allocated with SYNLVL (*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 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

**GET_ATTRIBUTES:** The GET_ATTRIBUTES verb requests information about the conversation that is attached to the program. This verb is supported by the get-attributes ICF operation. The values are returned to your record buffer; the length of the record buffer must be at least 144 bytes. To receive the entire record, the length of the record buffer must be at least 444 bytes. The values are returned in the order specified in Table C-1.

*Table C-1 (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 D U | The device is an ICF device type. The device is a display device. Unknown. |
| 32 through 37 | APPC | APPC communications type. |
| 38 | Y N | This is a requesting program device. This is a session acquired by a source program. |
| 39 | Y N | Program device has been acquired. Program device has not been acquired. |
| 40 | Y N | Input is invited for this program device. Input is not invited for this program device. |
| 41 | Y N | Invited input is available for this program device. Invited input is not available for this program device. |
| 42 through 50 | Reserved | Not applicable to communications. |

*Table C-1 (Page 2 of 3). Attribute Information Fields*

| Position | Value | Meaning |
|---|---|---|
| 51 | Y N | Session has an active transaction. Session does not have an active transaction. |
| 52[1] | 0 1 2 | Synchronization level is NONE. Synchronization level is CONFIRM. Synchronization level is COMMIT. |
| 53 | M B | Mapped conversation. 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 104 | Reserved | Not applicable to communications. |
| 105 | | APPC conversation state. |
| | X'00' | Reset. No conversation exists. |
| | X'01' | Send. Program can send data. |
| | X'02' | Defer receive. Program enters receive state after a confirm, flush, or commit operation completes successfully. |
| | X'03' | Defer deallocate. Program enters deallocate state after a commit operation completes successfully. |
| | X'04' | Receive. Program can receive data. |
| | X'05' | Confirm. Program received a confirmation request. |
| | X'06' | Confirm send. Program received a confirmation request and send control. |
| | X'07' | Confirm deallocate. Program received a confirmation request and deallocate notification. |
| | X'08' | Commit. Program received a commit request. |
| | X'09' | Commit send. Program received a commit request and send control. |
| | X'0A' | Commit deallocate. Program received a commit request and deallocate notification. |
| | X'0B' | Deallocate. Program received a deallocate notification. |
| | X'0C' | Rollback required. Program must roll back changes to protected resources. |
| 106 through 113 | Name | Conversation correlator. The conversation correlator associates the conversation states with the logical unit of work and is used during resynchronization. |
| 114 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 | ISDN unknown remote number plan. |
| | 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 190 | Character | ISDN remote number (blank padded EBCDIC). |
| 191 through 194 | Reserved | |

*Table C-1 (Page 3 of 3). Attribute Information Fields*

| Position | Value | Meaning |
|---|---|---|
| 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. |
| 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. |
| 391 | Binary | Length of the protected logical unit of work identifier (LUWID). Must be from 0 to 26. |
| 392 | Binary | Length of the qualified LU name. Must be from 0 to 17. |
| 393 through 409 | Character | Network-qualified protected LU name in the following form: *netid.luname*. *netid* is the network identifier. *luname* is the logical unit name. This field may be blank. |
| 410 through 415 | Character | Protected LUWID instance number. |
| 416 through 417 | Binary | Protected LUWID sequence number. |
| | | **Note:** The protected LUWID identifies the current logical unit of work for a protected conversation. |
| 418 | Binary | Length of the unprotected LUWID. Must be from 0 to 26. |
| 419 | Binary | Length of the qualified LU name. Must be from 0 to 17. |
| 420 through 436 | Character | Network-qualified unprotected LU name in the following form: *netid.luname*. *netid* is the network identifier. *luname* is the logical unit name. This field may be blank. |
| 437 through 442 | Character | Unprotected LUWID instance number. |
| 443 through 444 | Binary | Unprotected LUWID sequence number. |
| | | **Note:** The unprotected LUWID identifies the current logical unit of work for conversations with a synchronization level of none or confirm. |

**Note:**

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_FOR_SYNCPT:** The PREPARE_FOR_SYNCPT verb causes a single protected resource to be prepared to advance to the next synchronization point. Using this verb allows a program to have its own logic to respond if its partner cannot complete the commit process.

**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 I/O feedback area

**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)**
Use if conversation was allocated with SYNLVL(*COMMIT). It can also be used in place of TYPE(CONFIRM) for SYNLVL(*CONFIRM) conversations or in place of TYPE(FLUSH) for SYNLVL(*NONE) conversations.

- For data description specifications (DDS), use the TNSSYNLVL keyword. The TNSSYNLVL keyword is not allowed with an invite function if the conversation was allocated with SYNLVL(*COMMIT).
- Not supported for system-supplied formats.

For SYNLVL(*COMMIT), the conversation enters defer receive state. Defer receive state indicates that this program wants to be in receive state after a commit operation completes successfully. The only valid operations for the conversation in defer receive state are: commit, rollback, PRPCMT, and EOS.

**TYPE(CONFIRM)**
Use if conversation was allocated with SYNLVL (*CONFIRM).

- For DDS, use the CONFIRM keyword.
- Not supported for system-supplied formats.

**TYPE(FLUSH)**
Use if conversation was allocated with SYNLVL (*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)**
VARBUFMGT DDS keyword not specified. Implicit when using system-supplied formats.

**FILL(BUFFER)**
VARBUFMGT 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 following lists 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.

***ICF Return Codes Mapped to WHAT_RECEIVED Values:***
The following list shows ICF return codes with the corresponding WHAT_RECEIVED values for the RECEIVE_AND_WAIT verb.

**ICF Return Code WHAT_RECEIVED Values**

**0000** DATA[1], DATA_COMPLETE[2], SEND

**0001** DATA[1], DATA_COMPLETE[2]

**0008**[3] DATA[1], DATA_COMPLETE[2]

**0010** DATA[1], DATA_COMPLETE[2]

**0014** DATA[1], DATA_COMPLETE[2], CONFIRM_SEND

**0015** DATA[1], DATA_COMPLETE[2], CONFIRM

**001C**[3] DATA[1], DATA_COMPLETE[2], CONFIRM_DEALLOCATE

**0200** DATA[1], DATA_COMPLETE[2], SEND

**0201** DATA[1], DATA_COMPLETE[2]

**0208**[3] DATA[1], DATA_COMPLETE[2]

**0214** DATA[1], DATA_COMPLETE[2], CONFIRM_SEND

**0215** DATA[1], DATA_COMPLETE[2], CONFIRM

**021C**[3] DATA[1], DATA_COMPLETE[2], CONFIRM_DEALLOCATE

**0257** TAKE_COMMIT

**0258** TAKE_COMMIT_SEND

**0259** TAKE_COMMIT_DEALLOCATE

**0300** SEND

**0301** DATA[1], DATA_COMPLETE[2]

**0314** CONFIRM_SEND

**0315** CONFIRM

**031C** CONFIRM_DEALLOCATE

**0357** TAKE_COMMIT

**0358** TAKE_COMMIT_SEND

**0359** TAKE_COMMIT_DEALLOCATE

**3431** LL_TRUNCATED[2]

**3471** DATA_INCOMPLETE

[1] DATA is returned only when FILL(BUFFER) is used with VARBUFMGT.

[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.

***WHAT_RECEIVED Values Mapped to ICF Return Codes:***
The following list shows the WHAT_RECEIVED values for the RECEIVE_AND_WAIT verb with the corresponding ICF return codes.

**WHAT_RECEIVED Value ICF Return Codes**

**DATA**[1] 0000, 0001, 0008[3], 0010, 0014, 0015, 001C[3], 0200, 0201, 0208[3], 0214, 0215, 021C[3], 0301

**DATA_COMPLETE**[2] 0000, 0001, 0008[3], 0010, 0014, 0015, 001C[3], 0200, 0201, 0208[3], 0214, 0215, 021C[3], 0301

**DATA_INCOMPLETE** 3471

**LL_TRUNCATED**[2] 3431

**SEND** 0000, 0200, 0300,

**CONFIRM** 0015, 0215, 0315,

**CONFIRM_SEND** 0014, 0214, 0314

**CONFIRM_DEALLOCATE** 001C[3], 021C[3], 031C[3],

**TAKE_COMMIT** 0257, 0357

**TAKE_COMMIT_SEND** 0258, 0358

**TAKE_COMMIT_DEALLOCATE** 0259, 0359

[1] DATA is returned only when FILL(BUFFER) is used with VARBUFMGT.

[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.

***Response Indicators Mapped to WHAT_RECEIVED Values:*** The following list shows the DDS response indicators with the corresponding WHAT_RECEIVED values for the RECEIVE_AND_WAIT verb.

**Response Indicators WHAT_RECEIVED Values**

**RCVCONFIRM** CONFIRM, CONFIRM_SEND, CONFIRM_DEALLOCATE

**RCVTRNRND** SEND, CONFIRM_SEND, TAKE_COMMIT_SEND

**RCVDETACH** CONFIRM_DEALLOCATE, TAKE_COMMIT_DEALLOCATE

**RCVTKCMT** TAKE_COMMIT, TAKE_COMMIT_SEND, TAKE_COMMIT_DEALLOCATE

**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 VARBUFMGT keyword specified, implicit by the record format or specified explicitly by the VARLEN keyword.

For DDS without the VARBUFMGT 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.

If the conversation has a synchronization level of commit, the fail function may cause the system to roll back the protected LUW. If the fail function is in response to a commit operation, the system rolls back the protected LUW. If the fail function is in response to a prepare-for-commit function, the system does *not* roll back the protected LUW.

**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 Table C-1 on page C-11 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

**BACKOUT:** The BACKOUT verb is used to restore all protected resources to their status as of the last successful commit or rollback operation. This verb is supported by the rollback operation of the AS/400 system.

**RETURN_CODE**
Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

**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-attributes 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-attributes operation. Refer to Table C-1 on page C-11 for more information.

**SECURITY_USER_ID**
This is returned in positions 21 through 30 of the data returned by the ICF get-attributes operation. Refer to Table C-1 on page C-11 for more information.

**LUW_IDENTIFIER**
This is returned in positions 437 through 444 of the data returned by the ICF get-attributes operation. Refer to Table C-1 on page C-11 for more information.

**PROTECTED_LUW_IDENTIFIER**
This is returned in positions 410 through 417 of the data returned by the ICF get-attributes operation. Refer to Table C-1 on page C-11 for more information.

**GET_TYPE:** The GET_TYPE verb is used to determine the conversation type. This verb is supported by the get-attributes 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 Table C-1 on page C-11 for more information.

**SET_SYNCPT_OPTIONS:** The SET_SYNCPT_OPTIONS verb is used to change the options governing the processing of the SYNCPT, BACKOUT, PREPARE_FOR_SYNCPT, and MC_PREPARE_FOR_SYNCPT verbs. This verb is supported by the Change Commitment Options (QTNCHGCO) API. Changes made using the QTNCHGCO API only affect the activation group that it was called in. The *System API Programming* book has information about the QTNCHGCO API.

**RETURN_CODE**
Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

**VOTE_READ_ONLY_PERMITTED**

The vote read-only permitted parameter of the Change Commitment Options (QTNCHGCO) API is equivalent to this LU 6.2 parameter.

**WAIT_FOR_OUTCOME**

The wait for outcome parameter of the Change Commit-

ment Options (QTNCHGCO) API is equivalent to this LU 6.2 parameter.

**ACTION_IF_PROBLEMS**

The action if problems parameter of the Change Commitment Options (QTNCHGCO) API is equivalent to this LU 6.2 parameter.

**Note:** The QTNCHGCO API also has an action if ENDJOB parameter that allows you to specify what the system should do in the event of an ENDJOB.

**SYNCPT:** The SYNCPT verb is used to commit changes to all protected resources. This verb is supported by the commit operation of the AS/400 system.

**RETURN_CODE**
Information is returned as follows:

- Messages
- Return codes in the communications device dependent area of the I/O feedback area

**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

Table C-2 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.

*Table C-2 (Page 1 of 5). Mapping of LU 6.2 Return Codes to ICF Return Codes*

| LU 6.2 Return Code or Subcode | ICF Return Code | Description |
|---|---|---|
| ALLOCATION_ERROR | 82FA | Requested synchronization level not supported. |
| | 82FB | Protected conversations not supported on single-session connections. |
| | 82FC | Protected conversations not supported by System/36 and System/38 environments. |
| | 82FD | Exchange log name processing failed. |

*Table C-2 (Page 2 of 5). 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. |
| | 80FB | Abnormal end of session or remote protocol error. Rollback required. |
| | 8197 | Remote system ended transmission abnormally, session ended. |
| | 81F2 | Remote system ended transmission abnormally, session ended. Rollback required. |
| | 8327 | No active transaction. |
| DEALLOCATE_ABEND_PROG | 8197 | Abnormal end of session (program). |
| | 81F2 | Abnormal end of session (program). Rollback required. |
| | 83F1 | Abnormal end of session (program). |
| | 83FB | Abnormal end of session (program). Rollback required. |
| DEALLOCATE_ABEND_SVC | 81C5 | Abnormal end of session (service). |
| | 81F3 | Abnormal end of session (service). Rollback required. |
| DEALLOCATE_ABEND_TIMER | 81C6 | Abnormal end of session (timer). |
| | 81F4 | Abnormal end of session (timer). Rollback required. |
| DEALLOCATE_NORMAL | 0008 | Detach indication received with data. |
| | 0308 | Detach indication received without data. |
| MAP_EXECUTION_FAILURE | None | No mapping exists. |

*Table C-2 (Page 3 of 5). Mapping of LU 6.2 Return Codes to ICF Return Codes*

| LU 6.2 Return Code or Subcode | ICF Return Code | Description |
|---|---|---|
| MAP_NOT_FOUND | 83E0 | Record format was not defined for the ICF file. |
| MAPPING_NOT_SUPPORTED | None | No mapping exists. |
| OK | Major return codes 00, 02, 03, or 34 | Normal completion. |
| 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. |

*Table C-2 (Page 4 of 5). 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. |
| | 80FA | Device is not usable. Rollback required. |
| | 80FB | Session ended abnormally or remote protocol error. Rollback required. |
| | 8196 | Session was ended locally. |
| | 8197 | Remote system ended transmission abnormally, session ended. |
| | 81F1 | Session was ended locally. Rollback required. |
| | 81F2 | Remote system ended transmission abnormally, session ended. Rollback required. |
| RESOURCE_FAILURE_RETRY | 8081 | System error detected. |
| | 80C0 | Abnormal end of session or remote protocol error. |
| | 80F9 | System error detected. Rollback required. |
| | 80FB | Abnormal end of session or remote protocol error. Rollback required. |
| | 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. |
| | 81F0 | Permanent line or controller error. Rollback required. |
| | 81F1 | Session was ended locally. Rollback required. |
| | 81F2 | Remote system ended transmission abnormally, session ended. Rollback required. |
| | 81F5 | Data received does not match any record formats in file. Rollback required. |
| | 83F8 | Program device in error. |
| | 83FC | Program device in error. Rollback required. |
| ROLLBACK_REQUIRED | 0054 | Rollback required. |
| | 0254 | Rollback required. |
| ROLLED_BACK | 83FD | All affected protected resources rolled back. |
| | 83FE | Rollback results not yet known. |
| | 83FF | Transaction results mixed due to operator intervention. |
| SECURITY_NOT_VALID | 80EF | Open operation not authorized to user. |
| | 82EF | Acquire to device not authorized to user, or device is in service mode. |
| | 83CE | Security information specified not valid. |
| SYNC_LEVEL_NOT_SUPPORTED_BY_PGM | 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. |

*Table C-2 (Page 5 of 5). Mapping of LU 6.2 Return Codes to ICF Return Codes*

| LU 6.2 Return Code or Subcode | ICF Return Code | Description |
|---|---|---|
| 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 | None | No mapping exists. |

For information on return code values for mapped conversation verbs using the WHAT_RECEIVED parameter, see the lists under “MC_RECEIVE_AND_..WAIT” on page C-6. For information on return code values for basic conversation verbs using the WHAT_RECEIVED parameter, see the lists under “RECEIVE_AND_WAIT” on page C-14.

## 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 supports. 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. Table C-3 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.

*Table C-3 (Page 1 of 4). 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.<br>**Note:** Supported only by CPI Communications. |

*Table C-3 (Page 2 of 4). Supported Option Sets*

| Set Number | Set Name | Set Description |
|---|---|---|
| 108 | Sync point services | This option set allows a program to request sync point processing for all protected resources of the transaction. This option set includes the SYNCPT, BACKOUT, and SET_SYNCPT_OPTIONS verbs, all of which are type-independent verbs. This option set relates to the SYNC_LEVEL parameter of the [MC_]ALLOCATE, [MC_]DEALLOCATE, [MC_]GET_ATTRIBUTES, DEFINE_TP, and DISPLAY_TP verbs, to the PROTECTED_LUW_IDENTIFIER parameter of the GET_TP_PROPERTIES verb, and to the CONVERSATION_STATE parameter of the [MC_]GET_ATTRIBUTES verb. |
| **110** | Get conversation type | 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. |
| 111 | Recovery from program errors detected during sync point | This option set allows a program to issue the [MC_]PREPARE_FOR_SYNCPT verb to initiate a sync point operation on a conversation. If a program error return code is received after this verb is issued, the program can attempt to correct the problem and retry the sync point operation. This contrasts to the SYNCPT verb, which backs out the transaction if a program error is detected during the sync point operation. This option set includes the [MC_]PREPARE_FOR_SYNCPT verb and affects the return code of the BACKOUT verb. Option set 108 is a prerequisite. |
| 203 | Immediate allocation of a session | 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.<br><br>**Note:** Supported only by CPI Communications. |
| 204 | Conversations between programs located at the same LU | 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. |
| 211 | Session-level LU-LU verification | 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. |
| 212 | User ID verification | 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. |
| 213 | Program-supplied user ID and password | 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. |
| 214 | User ID authorization | 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. |

*Table C-3 (Page 3 of 4). Supported Option Sets*

| Set Number | Set Name | Set Description |
|---|---|---|
| 217 | Profile pass-through | 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. |
| 218 | Program-supplied profile | 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 or 224 is a prerequisite. |
| 222 | Receive sign-on transaction | This option set receives the sign-on transaction. It is only used to check the password independently of any user program. It can also be used to change the password. Option set 212 is a prerequisite. |
| 223 | User password protection | This option set allows a program to send passwords encrypted. It changes the passwords used in option sets 212, 213, 219, 220, 221 and 222. Option set 211 is a prerequisite. |
| 224 | Program required password protection | This option set forces the password to be encrypted. This option set relates to the SECURITY parameter of the [MC_]ALLOCATE verb. Option set 211 is a prerequisite.<br><br>**Note:** Supported only by CPI Communications. |
| 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.<br><br>**Note:** ILE 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.<br><br>**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. |
| 249 | Vote read-only response to a sync point operation | This option set improves performance of sync point operations by allowing the local LU to vote read-only when none of the protected resources in its part of the distributed transaction have been changed. This option set includes the VOTE_READ_ONLY_PERMITTED parameter of the SET_SYNCPT_OPTIONS verb, and affects the return code to the SYNCPT verb. Option set 108 is a prerequisite. |

*Table C-3 (Page 4 of 4). Supported Option Sets*

| Set Number | Set Name | Set Description |
|---|---|---|
| 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.<br>**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.

*Table C-4 (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.<br>**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:<br>• SEND_MAX_RU_SIZE_LOWER_BOUND<br>• SEND_MAX_RU_SIZE_UPPER_BOUND<br>• RECEIVE_MAX_RU_SIZE_LOWER_BOUND<br>• 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. |

*Table C-4 (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, non-switched)
- X.21 Short-Hold Mode communications between two AS/400 systems
- Defining controller descriptions for programs communicating on the same system

The *Communications Configuration* book has an example of configuring for APPC over TCP/IP support.

## Switched Network without APPN Support—Configuration Example

In this example, two AS/400 systems are being configured to communicate with each other using switched connections without APPN networking functions.

In this 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 CRTLINSDLC. The following displays show the creation of this line description.

```
                    Create
Line Desc (SDLC) (CRTLINSDLC)

 Type choices, press Enter.

 Line description . . . . . > LALINSW    Name
 Resource names . . . . . > LIN011    Name
           + for more values
 Online at IPL  . . . . . .  *YES          *YES, *NO
 Data link role . . . . . .  *NEG          *NEG, *PRI, *SEC
 Physical interface . . . . .  *RS232V24    *RS232V24, *V35, *X21, ...
 Connection type  . . . . . .> *SWTPP     *NONSWTPP, *SWTPP, *MP, *SHM
 Vary on wait . . . . . .  *NOWAIT       *NOWAIT, 15-180 (1 second)
 Autocall unit  . . . . . .  *NO          *NO, *YES
 Exchange identifier  . . . . . > 05600001 1   05600000-056FFFFF, *SYSGEN
 NRZI data encoding . . . . . .  *YES          *YES, *NO
 Line speed . . . . . .  9600   600, 1200, 2400, 4800...
 Modem type supported . . . . . .  *NORMAL     *NORMAL, *V54, *IBMWRAP...
 Switched connection type . . . .  *BOTH         *BOTH, *ANS, *DIAL
 Autoanswer . . . . . .  *YES          *YES, *NO
 Autodial . . . . . .  *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 Line Desc (SDLC)
(CRTLINSDLC)

 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         0-2048 (0.0001 seconds)
 Nonproductive receive timer  . .  320         160-4200 (0.1 seconds)
 Idle timer . . . . . .  30         5-300 (0.1 seconds)
 Connect poll timer . . . . . .  30         2-300 (0.1 seconds)
 Poll cycle pause . . . . . .  0         0-2048 (0.0001 seconds)
 Frame retry  . . . . . .  7         0-64
 Data Set Ready drop timer  . . .  6         3-60 (seconds)
 Autoanswer type  . . . . . .  *DTR         *DTR, *CDSTL
 Remote answer timer  . . . . . .  60         30, 35, 40, 45 (seconds)...
                                                                  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) (CRTLINSDLC)

 Type choices, press Enter.

 Text 'description' . . . . . > 'Line description for
NY to LA'
_
```

**Considerations for specifying the CRTLINSDLC 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 . . . . . > LACTLSW     Name
Link type  . . . . . .> *SDLC          *ANYNW, *FAX, *FR, *IDLC...
Online at IPL  . . . . . .   *YES           *YES, *NO
Switched connection  . . . . . > *YES         *NO, *YES
Short hold mode  . . . . . .   *NO            *NO, *YES
APPN-capable . . . . . > *NO          *YES, *NO
Controller type  . . . . . .   *BLANK         *BLANK, *FBSS, 3174, 3274...
Switched line list . . . . . > LALINSW 1      Name
            + for more values
Maximum frame size . . . . . .   *LINKTYPE      265-16393, 256, 265, 512...
Remote network identifier  . . .   *NETATR        Name, *NETATR, *NONE, *ANY
Remote control point . . . . . .                 Name, *ANY
Exchange identifier  . . . . . > 05600002 2    00000000-FFFFFFFF
Initial connection . . . . . .   *DIAL          *DIAL, *ANS
Dial initiation  . . . . . .   *LINKTYPE      *LINKTYPE, *IMMED, *DELAY
Connection number  . . . . . > 81355551234 3
                                                                    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.

Data link role . . . . . .   *NEG           *NEG, *PRI, *SEC
Station address  . . . . . > 01 4          00-FE
Autocreate device  . . . . . .   *ALL           *ALL, *NONE
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  . . . . . > LOSANGEL 1   Name
Online at IPL  . . . . . .   *YES           *YES, *NO
Local location . . . . . > NEWYORK 2     Name, *NETATR
Remote network identifier  . . .   *NETATR        Name, *NETATR, *NONE
Attached controller  . . . . . > LACTLSW     Name
Mode . . . . . > BLANK       Name, *NETATR
            + for more values > #BATCH
Message queue  . . . . . .   QSYSOPR     Name, QSYSOPR
  Library  . . . . . .   *LIBL      Name, *LIBL, *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, *VFYENCPWD
                                                                    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. The example shows the commands used in a CL program; the configuration can also be performed using the configuration displays shown previously.

```
/*******************************************************************/
/*                                                                 */
/*  MODULE:  LANYAPPC                  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) CNNNBR('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
```

## Nonswitched Network without APPN Support—Configuration Example

The commands shown here are those used to configure two AS/400 systems to communicate with each other without using APPN networking functions.

In this 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. 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 */
            CRTLINSDLC LIND(LOSANGEL) RSRCNAME(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) (CRTLINSDLC). The parameters specified are:

**LIND(LOSANGEL)**
The name assigned to the line description is LOSANGEL.

**RSRCNAME(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:

**DEVD(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 */
            CRTLINSDLC LIND(NEWYORK) RSRCNAME(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
```

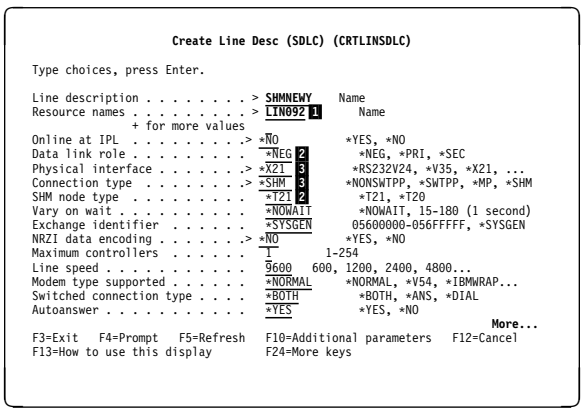
## X.21 Short-Hold Mode—Configuration Example

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) (CRTLINSDLC)

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 . . . . .  T1         *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 . . .  *CPS41, *CPS42, *CPS43...
            + 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 . . . . .> LOSANGEL    Name
Link type  . . . . . . . . . . .> *SDLC       *ANYNW, *FAX, *FR, *IDLC...
Online at IPL  . . . . . . . . .  *YES         *YES, *NO
Switched connection  . . . . . .> *YES 1      *NO, *YES
Short hold mode  . . . . . . . .> *YES 1      *NO, *YES
APPN-capable . . . . . . . . . .> *NO         *YES, *NO
Controller type  . . . . . . . .  *BLANK      *BLANK, *FBSS, 3174, 3274...
Switched line list . . . . . . .> SHMNEWY 2   Name
            + for more values
Maximum frame size . . . . . . .  *LINKTYPE   265-16393, 256, 265, 512...
Remote network identifier  . . .  *NETATR     Name, *NETATR, *NONE, *ANY
Remote control point . . . . . .> RCHAS045 3  Name, *ANY
Exchange identifier  . . . . . .              00000000-FFFFFFFF
Initial connection . . . . . . .  *DIAL       *DIAL, *ANS
Dial initiation  . . . . . . . .  *LINKTYPE   *LINKTYPE, *IMMED, *DELAY
Connection number  . . . . . . .> 31012981873 4
                                                                  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.

Data link role . . . . . . . . .  *NEG          *NEG, *PRI, *SEC
SHM disconnect limit . . . . . .  10            1-254, *NOMAX
SHM disconnect timer . . . . . .  50            2-3000 (0.1 seconds)
Station address  . . . . . . . .> C1            00-FE
Autocreate device  . . . . . . .  *ALL         *ALL, *NONE
Text 'description' . . . . . . .> 'APPC controller for SHM line to LOSANGEL'


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 CRTLINSDLC 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 . . . . . .> LOSANGEL    Name
Remote location  . . . . . . .> LOSANGEL 1    Name
Online at IPL  . . . . . . . .> *NO          *YES, *NO
Local location . . . . . . . .> NEWYORK 1     Name, *NETATR
Remote network identifier  . . .  *NETATR     Name, *NETATR, *NONE
Attached controller  . . . . .> LOSANGEL 2    Name
Mode . . . . . . . . . . . . .   *NETATR      Name, *NETATR
            + for more values
Message queue  . . . . . . . .   QSYSOPR     Name, QSYSOPR
  Library  . . . . . . . . . .     *LIBL     Name, *LIBL, *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, *VFYENCPWD
                                                                  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 CRTLINSDLC, CRTCTLAPPC, and CRTDEVAPPC commands are the same as for the NEWYORK system.

### Creating the Line Description

```
                    Create Line Desc (SDLC) (CRTLINSDLC)

Type choices, press Enter.

Line description . . . . . > SHMLOSA     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  . . . . . .> *SHM     *NONSWTPP, *SWTPP, *MP, *SHM
SHM node type  . . . . . .   *T21         *T21, *T20
Vary on wait . . . . . .     *NOWAIT      *NOWAIT, 15-180 (1 second)
Exchange identifier  . . . . .   *SYSGEN     05600000-056FFFFF, *SYSGEN
NRZI data encoding . . . . . .> *NO          *YES, *NO
Maximum controllers  . . . . .   1       1-254
Line speed . . . . . .       9600   600, 1200, 2400, 4800...
Modem type supported . . . . .   *NORMAL     *NORMAL, *V54, *IBMWRAP...
Switched connection type . . . .   *BOTH        *BOTH, *ANS, *DIAL
Autoanswer . . . . . .       *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) (CRTLINSDLC)

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 . . . . .   1        *NOMAX, 1-254 (0.1 seconds)
SHM call format  . . . . . .   *DNIC        *DNIC, *DCC
SHM access code  . . . . . .> 17         Character value
Calling number . . . . . .> 3101298187 3
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 . . .     *CPS41, *CPS42, *CPS43...
             + 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       *ANYNW, *FAX, *FR, *IDLC...
Online at IPL  . . . . . .> *NO          *YES, *NO
Switched connection  . . . . . > *YES         *NO, *YES
Short hold mode  . . . . . > *YES         *NO, *YES
APPN-capable . . . . . > *NO          *YES, *NO
Controller type  . . . . . .   *BLANK       *BLANK, *FBSS, 3174, 3274...
Switched line list . . . . . > SHMLOSA      Name
             + for more values
Maximum frame size . . . . . .   *LINKTYPE     265-16393, 256, 265, 512...
Remote network identifier  . . .   *NETATR      Name, *NETATR, *NONE, *ANY
Remote control point . . . . . > RCHAS320     Name
Exchange identifier  . . . . . .              00000000-FFFFFFFF
Initial connection . . . . . .   *DIAL        *DIAL, *ANS
Dial initiation  . . . . . .   *LINKTYPE     *LINKTYPE, *IMMED, *DELAY
Connection number  . . . . . > 31019764522
                                                                  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.

Data link role . . . . . .   *NEG          *NEG, *PRI, *SEC
SHM disconnect limit . . . . . .   10           1-254, *NOMAX
SHM disconnect timer . . . . . .   50           2-3000 (0.1 seconds)
Station address  . . . . . > C1           00-FE
Autocreate device  . . . . . .   *ALL         *ALL, *NONE
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  . . . . . > NEWYORK      Name
Online at IPL  . . . . . .> *NO          *YES, *NO
Local location . . . . . .> LOSANGEL     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  . . . . . .         *LIBL  Name, *LIBL, *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, *VFYENCPWD
                                                                  More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display       F24=More keys
```

## Programs Communicating on the Same System—Configuration Example

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. ICF Program Examples

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-2) shows two partner ILE C/400 programs.
- Example 2 (starting on page E-18) shows two partner COBOL/400 programs.
- Example 3 (starting on page E-34) shows two partner RPG/400 programs.

Each example contains two programs: the local program, which starts 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.

## 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:

```
CRTICFF 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 on page E-2.

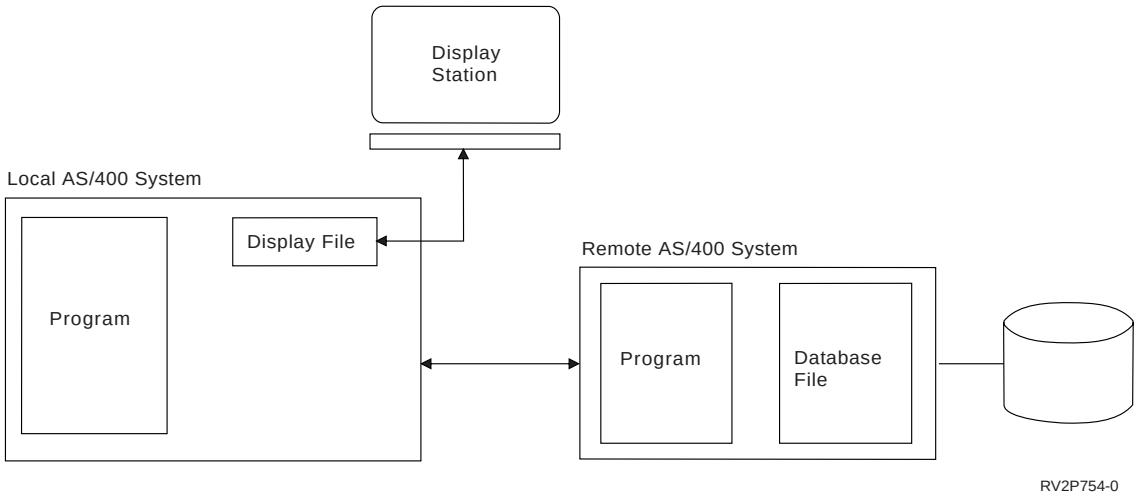



*Figure E-1. Inquiry Example*

```
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                                EVOKE(&PGMID);
A                                SECURITY(3 *USER)
A                                SYNLVL(*CONFIRM)
A        PGMID         10A  P
A*
A      R ITEMRQ
A                                CONFIRM
A                                ALWWRT
A        PARTNM         5A
A*
A      R ITEMDS
A                                ALWWRT
A        PARTDS        25A
A*
A      R ERRDES
A                                ALWWRT
A        ERRORD        40A
A*
A      R PGMEND
A                                DETACH
A*
A      R EOSREC
A                                EOS
A*
A      R PGMERR
A                                FAIL
A*
A      R ITEMOK
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/QATTDDS)
        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.

```
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  O 10 30
A   ERRORL 40A  O 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/QATTDDS) 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    ITEMD         25
A  K ITEMNM
```

*Figure E-4. DDS Source for the Database File*

## ILE C/400 Local Program for Inquiry Applications (Example 1)

The following explains the structure of the ILE 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 statement numbers in the program example illustrated in Figure E-5 on page E-5.

**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.

**Statement 79** 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.

**Statement 101** 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.

**Statement 122** The internal functions are prototyped so the ILE C/400 compiler knows the type of value returned and the type of parameters passed, if any.

**Statement 144** 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.

**Statement 151** 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.

**Statement 161** 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.

**Statement 170** 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.

**Statement 193** 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.

**Statement 196** The `cleanup` function is called to perform end-of-program processing.

**Statement 214** The `open_files` function opens the display and ICF files.

**Statement 230** 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.

**Statement 233** The separate indicator area for the display file T8189DSP is initialized.

**Statement 243** The `start_conversation` function establishes a conversation with the partner program.

**Statement 245** 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.

**Statement 250** 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, CPI Communications Program Examples.

**Statement 260** 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.

**Statement 277** 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 Programming* book for a description of the I/O feedback areas.

**Statement 325** The `cleanup` function performs end-of-program processing.

**Statement 328** 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.

**Statement 338** 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 (`_Rrelease` function) is then issued to detach the program from the session.

**Statement 345** The ICF and display files are closed.

```
                                        * * * * *  P R O L O G  * * * * *
 Program . . . . . :   T8189ICS
   Library . . . . . :     LAB
 Source file . . . . . :   QATTSYSC
   Library . . . . . :     QUSRTOOL
 Source member . . . . . :   T8189ICS
 Text Description  . . . . . :   APPC C program example ICF - Source
 Output  . . . . . :   *PRINT
 Compiler options  . . . . . :   *NOAGR   *NOEXPMAC   *LOGMSG   *NOSECLVL
                               :   *NOSHOWINC   *SHOWSKP   *NOXREF   *USRINCPATH
 Checkout options  . . . . . :   *NOACCURACY   *NOENUM   *NOEXTERN   *NOGENERAL   *NOGOTO   *NOINIT
                               :   *NOPARM   *NOPORT   *NOPPCHECK   *NOPPTRACE
 Optimization  . . . . . :   *NONE
 Debugging view  . . . . . :   *NONE
 Define names  . . . . . :
 Language level  . . . . . :   *SOURCE
 Source margins:
   Left margin . . . . . :   1
   Right margin  . . . . . :   32754
 Sequence columns:
   Left Column . . . . . :
   Right Column  . . . . . :
 Message flagging level  . . . . :   0
 Compiler messages:
   Message limit . . . . . :   *NOMAX
   Message limit severity  . . . :   30
 Replace module object . . . . . :   *YES
 User Profile  . . . . . :   *USER
 Authority . . . . . :   *LIBCRTAUT
 Target release  . . . . . :   *CURRENT
 System includes . . . . . :   *YES
 Last change . . . . . :   02/11/94 12:33:24
 Source description  . . . . . :   APPC C program example ICF - Source
 Compiler  . . . . . :   IBM ILE C/400 Compiler

                                        * * * * *  S O U R C E  * * * * *
 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
    1       |/*********************************************************************/
    2       |/* Program name.........: T8189ICS                                   */
    3       |/* Program description..: ICF local program                          */
    4       |/* Language.............: C/400                                      */
    5       |/*                                                                   */
    6       |/* This program invokes a program to handle part inquiry on          */
    7       |/* the remote system.  The acquire operation is used to              */
    8       |/* establish a communications session.  A write operation            */
    9       |/* with the evoke function is then 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 write operation        */
   14       |/* is issued (the data sent to the partner program is the            */
   15       |/* part number).  Note that the write operation was issued           */
   16       |/* with the confirm (CONFIRM) and allow-write (ALWWRT)               */
   17       |/* functions.  These functions results in the flushing of the        */
   18       |/* data (to be sent to the partner program), the changing of         */
   19       |/* the data flow direction (the partner program can send the         */
   20       |/* response), and the sending of a confirmation request to           */
   21       |/* the partner program.  If the partner program responds             */
   22       |/* with a positive response to the confirmation request (using       */
   23       |/* the RSPCONFIRM function), the ICF return code on the              */
   24       |/* write operation will be set to 0001 (indicating that              */
   25       |/* the part number was found); a read operation is then              */
   26       |/* issued to receive the part description.  However, if              */
   27       |/* the partner program responds with a negative response             */
   28       |/* to the confirmation request (using the FAIL function),            */
   29       |/* the ICF return code on the write operation will be set            */
   30       |/* to 83C9 (indicating that the part number was not found);          */
   31       |/* a read operation is issued to receive the error message.          */
   32       |/*                                                                   */
```

*Figure E-5 (Part 1 of 7). ILE C/400 Inquiry Example – Local Program*

```
   33       |/* The error message or part description (depending on              */
   34       |/* whether the part number was found) will be displayed on          */
   35       |/* the screen.                                                      */
   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             */
   53       |/*         the partner program is ready to receive data), the       */

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
   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 <recio.h>               /* Record i/o routines             */
   75       |
   76       |/********************************************************************/
   77       |/* Define the structures used for reads/writes from/to the ICF file. */
   78       |/********************************************************************/
   79       | struct {
   80       |   char pgmid??(10??);           /* target program name             */
   81       |} pgmstr_i_o = { "T8189ICT  " };
   82       |
   83       | struct {
   84       |   char partnm??(5??);           /* Part number                     */
   85       |} itemrq_i_o;
   86       |
   87       | struct {
   88       |   char partds??(25??);          /* part description                */
   89       |} itemds_i_o;
   90       |
   91       | struct {
   92       |   char errord??(40??);          /* error record                    */
   93       |} errdes_i_o;
   94       |char blank40??(40??) = "                                        ";
   95       |
   96       |
```

*Figure E-5 (Part 2 of 7). ILE C/400 Inquiry Example – Local Program*

```
   97       |/*********************************************************************/
   98       |/* Define the structures used for reads/writes from/to the display   */
   99       |/* file.                                                             */
  100       |/*********************************************************************/
  101       | struct {
  102       |   char partn??(5??);             /* part number                      */
  103       |   char partd??(25??);            /* part description                 */
  104       |   char errorl??(40??);           /* error record                     */
  105       |} prompt_i_o = { "     ", "                          ",
  106       |                 "                                     " };

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  107       |
  108       |/*********************************************************************/
  109       |/* Define constants/flags used in program.                           */
  110       |/*********************************************************************/
  111       |#define ERROR 1                  /* error during IO processing      */
  112       |#define NOERROR 0
  113       |#define MATCH 1                  /* ICF return code match indication*/
  114       |#define NO_MATCH 0
  115       |#define RC_0000 0                /* ICF return codes                */
  116       |#define RC_0001 1
  117       |#define RC_83C9 2
  118       |
  119       |/*********************************************************************/
  120       |/* Declare global variables/functions.                               */
  121       |/*********************************************************************/
  122       |int check_rc(int);
  123       |void cleanup(int);
  124       |void open_files(void);
  125       |void get_cust_num(void);
  126       |void start_conversation(void);
  127       |
  128       |_RFILE *icffptr;                  /* Pointer to the ICF file         */
  129       |_RFILE *dspfptr;                  /* Pointer to the display file     */
  130       |_XXIOFB_T *comm_fdbk;             /* IO Feedback for ICF unique info */
  131       |_XXIOFB_DSP_ICF_T *dsp_icf_fdbk;  /* IO Feedback - display & ICF file*/
  132       |_SYSindara dsp_indic;             /* indicator area for dsp          */
  133       |_SYSindara icf_indic;             /* indicator area for ICF          */
  134       |
  135       |
  136       |/*********************************************************************/
  137       |/* START OF PROGRAM                                                  */
  138       |/*                                                                   */
  139       |/* Files are opened, a conversation with the remote program is       */
  140       |/* started, and the part inquiry screen is displayed.  Inquiries     */
  141       |/* are handled until the user presses the F3=Exit key, in which case */
  142       |/* the conversation will be ended and the program will end.          */
  143       |/*********************************************************************/
  144       |main()
  145       |{
  146     1 | open_files();
  147     2 | start_conversation();
  148       |
  149     3 | get_cust_num();
  150       |
  151     4 | while (dsp_indic??(98??) != '1')
  152       |   {
  153       |
  154       |    /*****************************************************************/
  155       |    /* The part number that the user has requested information       */
  156       |    /* for is sent to the remote program using the write             */
  157       |    /* operation with the confirm and allow-write functions.         */
  158       |    /*****************************************************************/
  159     5 |    strncpy(itemrq_i_o.partnm, prompt_i_o.partn, 5);
```

*Figure E-5 (Part 3 of 7). ILE C/400 Inquiry Example – Local Program*

```
 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  160     6 |    _Rformat(icffptr, "ITEMRQ    ");
  161     7 |    _Rwrite(icffptr, &itemrq_i_o,; sizeof(itemrq_i_o));
  162       |
  163       |
  164       |    /*************************************************************/
  165       |    /* The read operation is issued to receive the response      */
  166       |    /* from the remote program (the response can either be       */
  167       |    /* an error message or the part description, depending       */
  168       |    /* on whether the part was found or not).                    */
  169       |    /*************************************************************/
  170     8 |    if (check_rc(RC_0001) == MATCH)
  171       |      {
  172     9 |       _Rformat(icffptr, "ITEMDS    ");
  173    10 |       _Rreadn(icffptr, &itemds_i_o,; sizeof(itemds_i_o), __DFT);
  174       |
  175    11 |       strncpy(prompt_i_o.partd, itemds_i_o.partds, 25);
  176    12 |       strncpy(prompt_i_o.errorl, blank40, 40);
  177       |      }
  178       |    else
  179    13 |      if (check_rc(RC_83C9) == MATCH)
  180       |        {
  181    14 |         _Rformat(icffptr, "ERRDES    ");
  182    15 |         _Rreadn(icffptr, &errdes_i_o,; sizeof(errdes_i_o), __DFT);
  183       |
  184    16 |         strncpy(prompt_i_o.errorl, errdes_i_o.errord, 40);
  185    17 |         strncpy(prompt_i_o.partd, "                        ", 25);
  186       |        }
  187       |      else                       /* Unexpected return code.         */
  188    18 |        cleanup(ERROR);
  189       |
  190    19 |    if (check_rc(RC_0000) == NO_MATCH)  /* Read operations ok?       */
  191    20 |      cleanup(ERROR);
  192       |
  193    21 |    get_cust_num();
  194       |   }                             /* end of while                    */
  195       |
  196    22 | cleanup(NOERROR);
  197       |
  198       |}                                /* end of MAIN routine             */
  199       |
  200       |/*********************************************************************/
  201       |/*                                                                   */
  202       |/*   ***************                                ***************   */
  203       |/*   *                   INTERNAL FUNCTIONS                      *   */
  204       |/*   ***************                                ***************   */
  205       |/*                                                                   */
  206       |/*********************************************************************/
  207       |
  208       |/*********************************************************************/
  209       |/* "OPEN_FILES" function                                             */
  210       |/*                                                                   */
  211       |/* This function opens the display and ICF files, and sets           */
  212       |/* indicator areas for each file.                                    */

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  213       |/*********************************************************************/
  214       |void open_files()
  215       |{
  216       | if ((dspfptr=_Ropen("T8189DSP","ar+ indicators=y riofb=y" ))
  217     1 |       == NULL)
  218       |   {
  219     2 |    printf("Display file failed to open.\n");
  220     3 |    exit(ERROR);
  221       |   }
  222       |
```

*Figure E-5 (Part 4 of 7). ILE C/400 Inquiry Example – Local Program*

```
  223       | if ((icffptr=_Ropen("T8189ICF","ar+ indicators=y riofb=y" ))
  224     4 |        == NULL)
  225       |   {
  226     5 |    printf("ICF file failed to open.\n");
  227     6 |    exit(ERROR);
  228       |   }
  229       |
  230     7 | _Rindara(dspfptr, dsp_indic);
  231     8 | _Rindara(icffptr, icf_indic);
  232       |
  233     9 | memset(dsp_indic, '0', 99);      /* Initialize indicator area.      */
  234       |}                                 /* end open_files...               */
  235       |
  236       |/*********************************************************************/
  237       |/* "START_CONVERSATION" function                                     */
  238       |/*                                                                   */
  239       |/* This function establishes a conversation with the remote system.  */
  240       |/* The program device is acquired, and an evoke request is issued    */
  241       |/* to start the program at the remote system.                        */
  242       |/*********************************************************************/
  243       |void start_conversation()
  244       |{
  245     1 | _Racquire(icffptr, "ICF00     ");
  246     2 | if (check_rc(RC_0000) == NO_MATCH)
  247     3 |   cleanup(ERROR);
  248       |
  249     4 | _Rformat(icffptr, "PGMSTR    ");
  250     5 | _Rwrite(icffptr, &pgmstr_i_o,; sizeof(pgmstr_i_o));
  251     6 | if (check_rc(RC_0000) == NO_MATCH)
  252     7 |   cleanup(ERROR);
  253       |}                                 /* end start_conversation...       */
  254       |
  255       |/*********************************************************************/
  256       |/* "GET_CUST_NUM" function                                           */
  257       |/*                                                                   */
  258       |/* Get a customer number from the display.                           */
  259       |/*********************************************************************/
  260       |void get_cust_num()
  261       |{
  262     1 | _Rformat(dspfptr,"PROMPT    ");
  263     2 | _Rwrite (dspfptr, &prompt_i_o,; sizeof(prompt_i_o));
  264     3 | memset(dsp_indic, '0', 99);
  265     4 | _Rreadn (dspfptr, &prompt_i_o,; sizeof(prompt_i_o), __DFT);

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  266       |}                                 /* end get_cust_num...             */
  267       |
  268       |/*********************************************************************/
  269       |/* "CHECK_RC" function                                               */
  270       |/*                                                                   */
  271       |/* This function compares the actual ICF return code received        */
  272       |/* on an operation with an ICF return code that was expected.        */
  273       |/* If the return codes match, then an indication that the return     */
  274       |/* codes matched is returned; otherwise, an error indication is      */
  275       |/* returned.                                                         */
  276       |/*********************************************************************/
  277       |check_rc(int rc_type)
  278       |{
  279     1 | comm_fdbk = _Riofbk(icffptr);
  280       | dsp_icf_fdbk = (_XXIOFB_DSP_ICF_T *) ((char *)  comm_fdbk +
  281     2 |                comm_fdbk->file_dep_fb_offset);
  282       |
```

*Figure E-5 (Part 5 of 7). ILE C/400 Inquiry Example – Local Program*

```
 283     3 | if (rc_type == RC_0000)
 284       |   {
 285       |    if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&;
 286     4 |        strncmp(dsp_icf_fdbk->minor_ret_code, "00", 2) == 0)
 287     5 |      return(MATCH);
 288       |    else
 289     6 |      return(NO_MATCH);
 290       |   }
 291       | else
 292     7 |   if (rc_type == RC_0001)
 293       |     {
 294       |      if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&;
 295     8 |          strncmp(dsp_icf_fdbk->minor_ret_code, "01", 2) == 0)
 296     9 |        return(MATCH);
 297       |      else
 298    10 |        return(NO_MATCH);
 299       |     }
 300       |   else
 301    11 |     if (rc_type == RC_83C9)
 302       |       {
 303       |        if (strncmp(dsp_icf_fdbk->major_ret_code, "83", 2) == 0 &&;
 304    12 |            strncmp(dsp_icf_fdbk->minor_ret_code, "C9", 2) == 0)
 305    13 |          return(MATCH);
 306       |        else
 307    14 |          return(NO_MATCH);
 308       |       }
 309       |     else
 310    15 |        return(NO_MATCH);
 311       |}                               /* end check_rc...                */
 312       |
 313       |
 314       |/*********************************************************************/
 315       |/* "CLEANUP" function.                                               */
 316       |/*                                                                   */
 317       |/* The following code handles the end-of-program processing.         */
 318       |/* This includes the ending of the conversation with                 */

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
 319       |/* the remote system (if conversation is active) by either           */
 320       |/* issuing a write operation with the detach function                */
 321       |/* followed by a release operation (for non-error conditions),       */
 322       |/* or by issuing a write operation with the end-of-session           */
 323       |/* function (for error conditions).                                  */
 324       |/*********************************************************************/
 325       |void cleanup(int endtype)
 326       |{
 327       |
 328       | if ((endtype == ERROR) &&;
 329       |     (strncmp(dsp_icf_fdbk->major_ret_code, "80", 2) != 0) &&;
 330       |     (strncmp(dsp_icf_fdbk->major_ret_code, "81", 2) != 0) &&;
 331     1 |     (strncmp(dsp_icf_fdbk->major_ret_code, "82", 2) != 0))
 332       |   {
 333     2 |    _Rformat(icffptr, "EOSREC    ");
 334     3 |    _Rwrite(icffptr, 0, 0);
 335       |   }
 336       | else
 337       |   {
 338     4 |    _Rformat(icffptr, "PGMEND    ");
 339     5 |    _Rwrite(icffptr, 0, 0);
 340       |
 341     6 |    _Rrelease(icffptr, "ICF00     ");
 342       |   }
 343       |
```

*Figure E-5 (Part 6 of 7). ILE C/400 Inquiry Example – Local Program*

```
  344       |
  345     7 | _Rclose (icffptr);
  346     8 | _Rclose (dspfptr);
  347       |
  348     9 | exit(endtype);
  349       |}                                  /* end cleanup...                 */
  350       |

                                        * * * * *  E N D  O F  S O U R C E  * * * * *
                                             * * * * *  I N C L U D E S  * * * * *
INCNBR  Include Name                    Last change       Actual Include Name
    1   stdio.h                         12/02/93 14:12:18  QCLE/H/STDIO
    2   stdlib.h                        12/02/93 14:12:19  QCLE/H/STDLIB
    3   string.h                        12/02/93 14:12:19  QCLE/H/STRING
    4   stddef.h                        12/02/93 14:12:17  QCLE/H/STDDEF
    5   xxfdbk.h                        12/02/93 14:12:23  QCLE/H/XXFDBK
    6   recio.h                         12/02/93 14:12:15  QCLE/H/RECIO
                                      * * * * *  E N D  O F  I N C L U D E S  * * * * *
                                       * * * * *  M E S S A G E  S U M M A R Y  * * * * *
       Total           Informational(00)          Warning(10)          Error(30)           Severe Error(40)
         0                    0                       0                    0                    0
                              * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * * * *
Program T8189ICS was created in library LAB on 02/11/94 at 12:41:43.
                                 * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

*Figure E-5 (Part 7 of 7). ILE C/400 Inquiry Example – Local Program*

## ILE C/400 Remote Program for Inquiry Applications (Example 1)

The following explains the structure of the ILE C/400 remote program that handles requests sent by the partner program.

## Program Explanation

The reference numbers in the explanation below correspond to the statement numbers in the program example illustrated in Figure E-6 on page E-13.

**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.

**Statement 68** This structure defines the database file (T8189DB) structures used in the program. T8189DB is the database file used to read the customer records.

**Statement 76** 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.

**Statement 106** The internal functions are prototyped so the ILE C/400 compiler knows the type of value returned and the type of parameters passed, if any.

**Statement 138** 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.

**Statement 141** The program loops until there are no more requests to process, or an error occurs in the transaction with the partner program.

**Statement 149** A search of the database file is performed using the part number received from the partner program as the key.

**Statement 159** 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.

**Statement 174** 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.

**Statement 187** The `cleanup` function is called to perform end-of-program processing.

**Statement 204** The `open_files` function opens the database and ICF files.

**Statement 220** 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.

**Statement 230** 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.

**Statement 243** 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.

**Statement 262** 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.

**Statement 315** The `cleanup` function performs end-of-program processing.

**Statement 317** If the ICF return code is 0308, indicating that a detach was received, a release operation (`_Rrelease` function) is issued to detach the program from the session.

**Statement 322** 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.

**Statement 330** The ICF and database files are closed.

```
                                        * * * * *  P R O L O G  * * * * *
 Program . . . . . :   T8189ICT
   Library . . . . . :     LAB
 Source file . . . . . :   QATTSYSC
   Library . . . . . :     QUSRTOOL
 Source member . . . . . :   T8189ICT
 Text Description  . . . . . :   APPC C program example ICF - Target
 Output  . . . . . :   *PRINT
 Compiler options  . . . . . :   *NOAGR   *NOEXPMAC   *LOGMSG   *NOSECLVL
                               :   *NOSHOWINC   *SHOWSKP   *NOXREF   *USRINCPATH
 Checkout options  . . . . . :   *NOACCURACY   *NOENUM   *NOEXTERN   *NOGENERAL   *NOGOTO   *NOINIT
                               :   *NOPARM   *NOPORT   *NOPPCHECK   *NOPPTRACE
 Optimization  . . . . . :   *NONE
 Debugging view  . . . . . :   *NONE
 Define names  . . . . . :
 Language level  . . . . . :   *SOURCE
 Source margins:
   Left margin . . . . . :   1
   Right margin  . . . . . :   32754
 Sequence columns:
   Left Column . . . . . :
   Right Column  . . . . . :
 Message flagging level  . . . . :   0
 Compiler messages:
   Message limit . . . . . :   *NOMAX
   Message limit severity  . . . :   30
 Replace module object . . . . . :   *YES
 User Profile  . . . . . :   *USER
 Authority . . . . . :   *LIBCRTAUT
 Target release  . . . . . :   *CURRENT
 System includes . . . . . :   *YES
 Last change . . . . . :   02/11/94 12:33:31
 Source description  . . . . . :   APPC C program example ICF - Target
 Compiler  . . . . . :   IBM ILE C/400 Compiler

                                        * * * * *  S O U R C E  * * * * *
 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
    1       |/*********************************************************************/
    2       |/* Program name.........: T8189ICT                                   */
    3       |/* Program description..: ICF remote program                         */
    4       |/* Language.............: C/400                                      */
    5       |/*                                                                   */
    6       |/* This program accepts the incoming conversation by issuing         */
    7       |/* the acquire operation to acquire the requesting program           */
    8       |/* device.  A read operation is then issued to receive the           */
    9       |/* part number from the remote system.  The read operation           */
   10       |/* completes with an ICF return code of 0014, indicating             */
   11       |/* that data with a turnaround indication was received, and          */
   12       |/* that the partner program also requested confirmation.  The        */
   13       |/* database file T8189DB   is searched for the received part         */
   14       |/* number.  If the part number is found, a write operation           */
   15       |/* with the respond-to-confirm (RSPCONFIRM) function is              */
   16       |/* issued, followed by a write operation containing the              */
   17       |/* part description corresponding to the part number                 */
   18       |/* retrieved from the database file.  However, if the part           */
   19       |/* number is not found, a write operation with the                   */
   20       |/* negative-response (FAIL) function is issued, followed by          */
   21       |/* a write operation containing an error message describing          */
   22       |/* the error.  The write operation sending either the part           */
   23       |/* description or the error message is issued with the               */
   24       |/* allow-write (ALWWRT) function.  Using the allow-write             */
   25       |/* function results in the flushing of the data and the              */
   26       |/* changing of the data flow direction.  The partner program         */
   27       |/* can send more inquiries.                                          */
   28       |/*                                                                   */
   29       |/* This program will continue to handle inquiries from the           */
   30       |/* partner program until a detach indication is received.            */
   31       |/* Then the program ends.                                            */
   32       |/*                                                                   */
```

*Figure E-6 (Part 1 of 6). ILE C/400 Inquiry Example – Remote Program*

```
   33       |/* NOTE 1: If an unexpected ICF return code is received on          */
   34       |/*         any of the read or write operations, the                 */
   35       |/*         program will abnormally end the conversation (if         */
   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       |/*                                                                  */
   53       |/*******************************************************************/

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
   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 <recio.h>               /* record i/o routines             */
   64       |
   65       |/*******************************************************************/
   66       |/* Define the structure used for reads from the database file.       */
   67       |/*******************************************************************/
   68       | struct          {
   69       |   char partn??(5??);
   70       |   char partd??(25??);
   71       |} part_rec;
   72       |
   73       |/*******************************************************************/
   74       |/* Define the structures used for reads/writes from/to the ICF file. */
   75       |/*******************************************************************/
   76       | struct errdes {
   77       |   char errord??(40??);
   78       |} errdes_i_o;
   79       |
   80       |struct {
   81       |   char partnm??(5??);
   82       |} itemrq;
   83       |
   84       |struct {
   85       |   char partds??(25??);
   86       |} itemds;
   87       |
   88       |
   89       |/*******************************************************************/
   90       |/* Define constants/flags used in program.                         */
   91       |/*******************************************************************/
   92       |#define ERROR 1                 /* error during I/O processing      */
   93       |#define NOERROR 0
   94       |#define MATCH 1                 /* ICF return code match indication */
   95       |#define NO_MATCH 0
   96       |#define MORE_REQUESTS 0         /* More request indicator           */
   97       |#define NO_REQUESTS 1
   98       |#define RC_0000 0               /* ICF return codes                 */
   99       |#define RC_0001 1
  100       |#define RC_0308 2
```

*Figure E-6 (Part 2 of 6). ILE C/400 Inquiry Example – Remote Program*

```
  101       |#define RC_0014 3
  102       |
  103       |/*********************************************************************/
  104       |/* Declare global variables/functions.                               */
  105       |/*********************************************************************/
  106       |int check_rc(int);

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  107       |int get_cust_num(void);
  108       |void cleanup(void);
  109       |void open_files(void);
  110       |void start_conversation(void);
  111       |
  112       |char part_not_found??(40??) =
  113       |                              "THE REQUESTED PART WAS NOT FOUND  ";
  114       |
  115       |_RFILE *icffptr;                /* Pointer to the ICF file          */
  116       |_RFILE *dbfptr;                 /* Pointer to database file.        */
  117       |
  118       |_XXIOFB_T *comm_fdbk;           /* IO Feedback for ICF unique info  */
  119       |_XXIOFB_DSP_ICF_T *dsp_icf_fdbk; /* IO feedback - display & ICF file */
  120       |
  121       |_RIOFB_T *db_fdbk;              /* IO Feedback - data base file     */
  122       |
  123       |_SYSindara icf_indic;           /* indicatior area for ICF
  124       |size_t           size;          /* "size_t" is a synonym for the    */
  125       |                                /* type of the value returned by    */
  126       |                                /* the "sizeof" operator.           */
  127       |
  128       |
  129       |/*********************************************************************/
  130       |/* START OF PROGRAM                                                  */
  131       |/*                                                                   */
  132       |/* Files are opened, a conversation with the remote program is       */
  133       |/* started, and the part inquiry processing starts.  Inquiries       */
  134       |/* are handled until a detach indication is received.                */
  135       |/*********************************************************************/
  136       |main()
  137       |{
  138     1 | open_files();
  139     2 | start_conversation();
  140       |
  141     3 | while (get_cust_num() != NO_REQUESTS)
  142       |   {
  143       |    /*****************************************************************/
  144       |    /* A search of the database file is done using the part          */
  145       |    /* number as the key.                                            */
  146       |    /*****************************************************************/
  147     4 |    strncpy (part_rec.partn,itemrq.partnm,5);
  148     5 |    strncpy (part_rec.partd,"                          ",25);
  149       |    db_fdbk = _Rreadk(dbfptr, &part_rec,; sizeof(part_rec),
  150     6 |              __KEY_EQ, &part_rec;partn, sizeof(part_rec.partn));
  151       |
  152       |    /*****************************************************************/
  153       |    /* If the part number is found, a positive response to the       */
  154       |    /* confirmation request is issued, followed by a write           */
  155       |    /* operation with the requested information.  Otherwise,         */
  156       |    /* a negative response to the confirmation request is issued,    */
  157       |    /* followed by a write operation with an error message.          */
  158       |    /*****************************************************************/
  159     7 |    if (db_fdbk -> num_bytes > 0)    /* if record was found          */

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  160       |      {
  161     8 |       _Rformat(icffptr, "ITEMOK    ");
  162     9 |       _Rwrite(icffptr, 0, 0);
  163    10 |       if (check_rc(RC_0000) == NO_MATCH)
  164    11 |         cleanup();
```

*Figure E-6 (Part 3 of 6). ILE C/400 Inquiry Example – Remote Program*

```
 165       |
 166    12 |       strncpy(itemds.partds,part_rec.partd,25);
 167    13 |       _Rformat(icffptr, "ITEMDS    ");
 168    14 |       _Rwrite(icffptr, &itemds,; sizeof(itemds));
 169    15 |       if (check_rc(RC_0001) == NO_MATCH)
 170    16 |         cleanup();
 171       |      }
 172       |    else                            /* part description not found.  */
 173       |      {
 174    17 |       _Rformat(icffptr, "PGMERR    ");
 175    18 |       _Rwrite(icffptr, 0 ,0);
 176    19 |       if (check_rc(RC_0000) == NO_MATCH)
 177    20 |         cleanup();
 178       |
 179    21 |       strncpy(errdes_i_o.errord, part_not_found, 40);
 180    22 |       _Rformat(icffptr, "ERRDES    ");
 181    23 |       _Rwrite(icffptr, &errdes_i_o,; sizeof(errdes_i_o));
 182    24 |       if (check_rc(RC_0001) == NO_MATCH)
 183    25 |         cleanup();
 184       |      }
 185       |   }                                /* end WHILE                    */
 186       |
 187    26 | cleanup();
 188       |}                                   /* end of main routine          */
 189       |
 190       |/*********************************************************************/
 191       |/*                                                                   */
 192       |/*   ***************                                ***************   */
 193       |/*   *                   INTERNAL FUNCTIONS                      *   */
 194       |/*   ***************                                ***************   */
 195       |/*                                                                   */
 196       |/*********************************************************************/
 197       |
 198       |/*********************************************************************/
 199       |/* "OPEN_FILES" function                                             */
 200       |/*                                                                   */
 201       |/* This function opens the database and ICF files, and sets the      */
 202       |/* indicator area for the ICF file.                                  */
 203       |/*********************************************************************/
 204       |void open_files()
 205       |{
 206       |
 207       | if ((icffptr= _Ropen("T8189ICF","ar+ indicators=y riofb=y" ))
 208     1 |       == NULL)
 209       |   {
 210     2 |    printf("ICF file failed to open.\n");
 211     3 |    exit(ERROR);
 212       |   }

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
 213       |
 214     4 | if ((dbfptr= _Ropen("T8189DB", "rr riofb=n")) == NULL)
 215       |   {
 216     5 |    printf("Data Base file failed to open.\n");
 217     6 |    exit(ERROR);
 218       |   }
 219       |
 220     7 | _Rindara(icffptr, icf_indic);
 221       |
 222       |}                                   /* end open_files...            */
 223       |
 224       |/*********************************************************************/
 225       |/* "START_CONVERSATION" function                                     */
 226       |/*                                                                   */
 227       |/* This function establishes a conversation with the remote system.  */
 228       |/* The "requesting" program device is acquired.                      */
 229       |/*********************************************************************/
```

*Figure E-6 (Part 4 of 6). ILE C/400 Inquiry Example – Remote Program*

```
230       |void start_conversation()
  231       |{
  232     1 | _Racquire(icffptr, "ICF01     ");
  233     2 | if (check_rc(RC_0000) == NO_MATCH)
  234     3 |   cleanup();
  235       |}                                /* end start_conversation...       */
  236       |
  237       |/*********************************************************************/
  238       |/* "GET_CUST_NUM" function                                           */
  239       |/*                                                                   */
  240       |/* This subroutine waits for incoming data from the partner          */
  241       |/* program by issuing the read operation.                            */
  242       |/*********************************************************************/
  243       |get_cust_num()
  244       |{
  245     1 | _Rformat(icffptr, "ITEMRQ    ");
  246     2 | _Rreadn(icffptr, &itemrq,; sizeof(itemrq), __DFT);
  247     3 | if (check_rc(RC_0014) == MATCH)
  248     4 |   return(MORE_REQUESTS);
  249       | else
  250     5 |   return(NO_REQUESTS);
  251       |}                                /* end get_cust_num...             */
  252       |
  253       |/*********************************************************************/
  254       |/* "CHECK_RC" function                                               */
  255       |/*                                                                   */
  256       |/* This function compares the actual ICF return code received        */
  257       |/* on an operation with an ICF return code that was expected.        */
  258       |/* If the return codes match, then an indication that the return     */
  259       |/* codes matched is returned; otherwise, an indication that the      */
  260       |/* return codes did not match will be returned.                      */
  261       |/*********************************************************************/
  262       |check_rc(int rc_type)
  263       |{
  264     1 | comm_fdbk = _Riofbk(icffptr);
  265       | dsp_icf_fdbk = (_XXIOFB_DSP_ICF_T *) ((char *)  comm_fdbk +

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  266     2 |                comm_fdbk->file_dep_fb_offset);
  267       |
  268     3 | if (rc_type == RC_0000)
  269       |   {
  270       |    if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&;
  271     4 |        strncmp(dsp_icf_fdbk->minor_ret_code, "00", 2) == 0)
  272     5 |      return(MATCH);
  273       |    else
  274     6 |      return(NO_MATCH);
  275       |   }
  276       | else
  277     7 |   if (rc_type == RC_0001)
  278       |     {
  279       |      if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&;
  280     8 |          strncmp(dsp_icf_fdbk->minor_ret_code, "01", 2) == 0)
  281     9 |        return(MATCH);
  282       |      else
  283    10 |        return(NO_MATCH);
  284       |     }
  285       |   else
  286    11 |     if (rc_type == RC_0308)
  287       |       {
  288       |        if (strncmp(dsp_icf_fdbk->major_ret_code, "03", 2) == 0 &&;
  289    12 |            strncmp(dsp_icf_fdbk->minor_ret_code, "08", 2) == 0)
  290    13 |          return(MATCH);
  291       |        else
  292    14 |          return(NO_MATCH);
  293       |       }
```

*Figure E-6 (Part 5 of 6). ILE C/400 Inquiry Example – Remote Program*

```
  294       |     else
  295    15 |       if (rc_type == RC_0014)
  296       |         {
  297       |          if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0 &&;
  298    16 |              strncmp(dsp_icf_fdbk->minor_ret_code, "14", 2) == 0)
  299    17 |            return(MATCH);
  300       |          else
  301    18 |            return(NO_MATCH);
  302       |         }
  303       |       else
  304    19 |          return(NO_MATCH);
  305       |}                              /* end check_rc...               */
  306       |
  307       |/*********************************************************************/
  308       |/* "CLEANUP" function.                                               */
  309       |/*                                                                   */
  310       |/* The following code handles the end-of-program processing.         */
  311       |/* This includes the ending of the conversation with                 */
  312       |/* the remote system (if conversation is active), and the            */
  313       |/* closing of opened files.                                          */
  314       |/*********************************************************************/
  315       |void cleanup()
  316       |{
  317     1 | if (check_rc(RC_0308) == MATCH)
  318       |   {

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  319     2 |    _Rrelease(icffptr, "ICF01     ");
  320       |   }
  321       | else
  322       |   if ((strncmp(dsp_icf_fdbk->major_ret_code, "80", 2) != 0) &&;
  323       |       (strncmp(dsp_icf_fdbk->major_ret_code, "81", 2) != 0) &&;
  324     3 |       (strncmp(dsp_icf_fdbk->major_ret_code, "82", 2) != 0))
  325       |     {
  326     4 |      _Rformat(icffptr, "EOSREC    ");
  327     5 |      _Rwrite(icffptr, 0, 0);
  328       |     }
  329       |
  330     6 | _Rclose(icffptr);
  331     7 | _Rclose(dbfptr);
  332       |
  333     8 | exit(0);
  334       |}                                    /* end cleanup...           */
  335       |

                                    * * * * *  E N D  O F  S O U R C E  * * * * *
                                        * * * * *  I N C L U D E S  * * * * *
INCNBR  Include Name                    Last change        Actual Include Name
    1   stdio.h                         12/02/93 14:12:18  QCLE/H/STDIO
    2   stdlib.h                        12/02/93 14:12:19  QCLE/H/STDLIB
    3   string.h                        12/02/93 14:12:19  QCLE/H/STRING
    4   stddef.h                        12/02/93 14:12:17  QCLE/H/STDDEF
    5   xxfdbk.h                        12/02/93 14:12:23  QCLE/H/XXFDBK
    6   recio.h                         12/02/93 14:12:15  QCLE/H/RECIO
                                   * * * * *  E N D  O F  I N C L U D E S  * * * * *
                                    * * * * *  M E S S A G E  S U M M A R Y  * * * * *
       Total           Informational(00)         Warning(10)         Error(30)          Severe Error(40)
         0                  0                       0                  0                     0
                            * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * * * *
Program T8189ICT was created in library LAB on 02/11/94 at 12:42:30.
                               * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

*Figure E-6 (Part 6 of 6). ILE 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-21.

**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 Programming* book 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, CPI Communications Program Examples.

**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 . . . . . . . . . . . . :   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.  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).                                  <-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-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-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
  32 +000019       05  ERRDES        REDEFINES T8189ICF-RECORD.                         <-ALL-FMTS
  33 +000020           06 ERRORD                PIC X(40).                                <-ALL-FMTS
     +000021*    I-O FORMAT:PGMEND     FROM FILE T8189ICF   OF LIBRARY APPCLIB           <-ALL-FMTS
     +000022*                                                                             <-ALL-FMTS
     +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
      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-O 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-O 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
17      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 APPCREC FORMAT IS "PGMEND"
   120  003150         DROP "ICF00   " FROM T8189ICF
        003160       ELSE
19 121 003170         WRITE APPCREC 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-29.

**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, APPC 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 . . . . . :   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.  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 (ALWWRT) 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-O      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 ERRORD               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 ITEMNM               PIC X(5).                                <-ALL-FMTS
   41 +000007           06 ITEMD                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 APPCREC 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 ITEMD TO PARTDS OF ITEMDS
   85 002230       WRITE APPCREC FORMAT IS "ITEMDS"
      002240             TERMINAL IS PGM-DEV-NME
      002250     ELSE
11 86 002260       MOVE NOT-FND-MSG TO ERRORD OF ERRDES
   87 002270       WRITE APPCREC FORMAT IS "PGMERR"
      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 APPCREC 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-36.

**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 Programming* book 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, " CPI Communications Program Examples" 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  . . . . . :   *PGM
Text not available for message RXT0073 file QRPGMSG.
   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 . . . . . :   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
                     S o u r c e   L i s t i n g
      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 ERRORD
 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  IFEEDBK      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                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  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 (ALWWRT)       *
     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                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
     770  ILCLVAR      DS
     780  I                                    B   1   40REQLEN
     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                    EXFMTPROMPT
     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        REQLEN                      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        REQLEN                      003
    1190  C                      READ ERRDES                  88          3  003
    1200  C                      MOVE ERRORD    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 REQLEN                                002
SEQUENCE                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
    1290  C                      GOTO ENDCNV                                002
    1300  C                      END                                       E002
    1310  C*
    1320  C   6                EXFMTPROMPT                                 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                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  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.
           * * * * *   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.
* 7089        20   RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE T8189DSP.
* 7089        30   RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE T8189ICF.
                        C r o s s   R e f e r e n c e
 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
          ITEMDS                      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)        90D    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-43.

**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 QRPGMSG.
   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    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
                      S o u r c e   L i s t i n g
      10  H
1     20  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 ITEMNM
  A000002                                          6  30 ITEMD
  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 ERRORD
  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  IFEEDBK      DS
      90  I                                    B 372 3750ACTLEN
     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    *
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 (ALWWRT) 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            *
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                WRITEITEMOK                                       001
    1010  C           MAJMIN    CABNE'0000'    ENDCNV                             001
    1020  C   98                MOVELMSG,1     ERRORD                             001
    1030  C   98                WRITEERRDES                                       001
    1040  C  N98                MOVELITEMD     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
SEQUENCE                                                              IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  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
 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 ERRDES FILE T8189ICF FORMAT ERRDES.
  K000001                          ERRORD    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.
           * * * * *  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.
* 7089        30  RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE T8189ICF.
SEQUENCE                                                                                                  LAST
 NUMBER    *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8             UPDATE
                  C o m p i l e - T i m e  T a b l e s
 Table/Array  . . . . . :   MSG
    1820  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
                     PHYSICAL    LOGICAL
       FILE/RCD      FIELD       FIELD       ATTRIBUTES
   01  T8189DB
         DBRCD
                     ITEMNM                  CHAR    5
                       C r o s s  R e f e r e n c e
 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
        ERRORD      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
 Indicator References:
        INDICATOR  REFERENCES (M=MODIFIED D=DEFINED)
        LR            1340M
* 7031  88            1290M    1630M
        98             880M     980     1000     1020     1030     1040
                      1050
      * * * * *  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:    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.
      * * * * *  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
              4       4      0      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.
          * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

*Figure E-10 (Part 5 of 5). RPG/400 Inquiry Example – Remote Program*

# Appendix F. CPI Communications Program Examples

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 ILE C/400 programs.
- Example 2 (starting on page F-16) shows two COBOL/400 programs.
- Example 3 (starting on page F-33) 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 on page F-2 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.

## 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/QATTDDS)
        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                                       5 10'Part Number: '
A          PARTN          5A  B  5 25
A                                      10 10'Part Description: '
A          PARTD         25A  O 10 30
A          ERRORL        40A  O 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/QATTDDS) 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           ITEMD          25
A         K ITEMNM
```

*Figure F-3. DDS Source for the Database File*

## ILE C/400 Local Program for Inquiry Applications (Example 1)

The following explains the structure of the ILE 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 statement numbers in the program example illustrated in Figure F-4 on page F-4.

**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.

**Statement 75** The preprocessor include directive `#include "QSYSINC/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.

**Statement 99** 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.

**Statement 115** The internal functions are prototyped so the ILE C/400 compiler knows the type of value returned and the type of parameters passed, if any.

**Statement 136** 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.

**Statement 141** 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

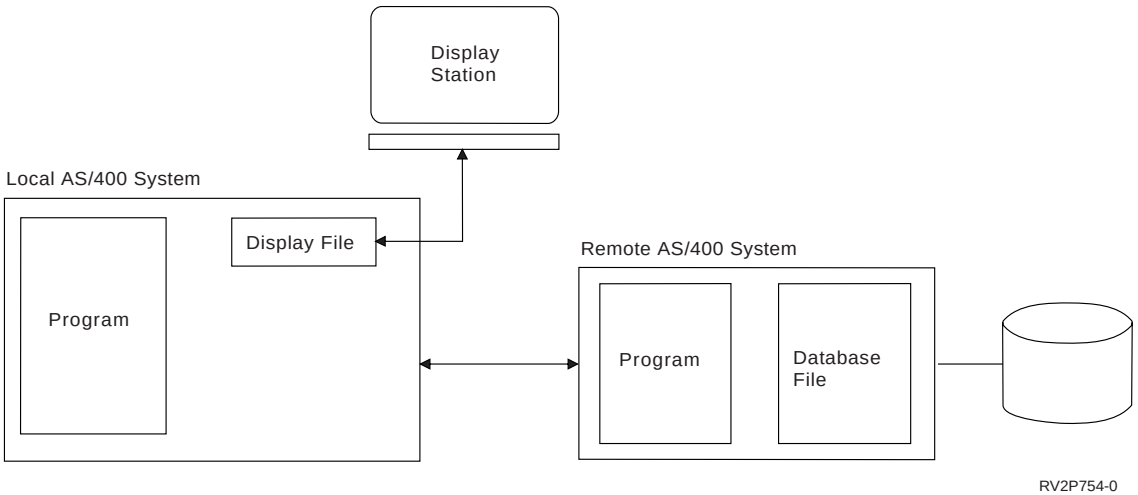



*Figure F-1. Inquiry Example*

an error occurs in the transaction with the partner program.

**Statement 149** 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.

**Statement 158** 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.

**Statement 185** 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.

**Statement 188** The `cleanup` function is called to perform end-of-program processing.

**Statement 205** The `open_files` function opens the display file.

**Statement 214** 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.

**Statement 216** The separate indicator area for the display file T8189DSP is initialized.

**Statement 225** The `start_conversation` function establishes a conversation with the partner program and sets various conversation characteristics.

**Statement 232** 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.

**Statement 242** 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, " ICF Program Examples" on page E-1.

**Statement 249** The Set_Sync_Level (CMSSL) call is issued so that the *sync_level* conversation characteristic is set to CM_CONFIRM.

**Statement 255** The Allocate (CMALLC) call is issued so that a conversation can be started using the *conversation_ID* previously assigned by the CMINIT call.

**Statement 265** The Set_Send_Type (CMSST) call is issued so that the *send_type* conversation characteristic is set to CM_SEND_AND_PREP_TO_RECEIVE.

**Statement 274** 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.

**Statement 292** The `cleanup` function performs end-of-program processing.

**Statement 294** 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.

**Statement 301** 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.

**Statement 314** The display file is closed.

```
                                        * * * * *  P R O L O G  * * * * *
 Program . . . . . . . . . . . . :   T8189CCS
   Library . . . . . . . . . . . :     LAB
 Source file . . . . . . . . . . :   QATTSYSC
   Library . . . . . . . . . . . :     QUSRTOOL
 Source member . . . . . . . . . :   T8189CCS
 Text Description  . . . . . . . :   APPC C program example CPIC - Source
 Output  . . . . . . . . . . . . :   *PRINT
 Compiler options  . . . . . . . :   *NOAGR   *NOEXPMAC   *LOGMSG   *NOSECLVL
                                 :   *NOSHOWINC   *SHOWSKP   *NOXREF   *USRINCPATH
 Checkout options  . . . . . . . :   *NOACCURACY   *NOENUM   *NOEXTERN   *NOGENERAL   *NOGOTO   *NOINIT
                                 :   *NOPARM   *NOPORT   *NOPPCHECK   *NOPPTRACE
 Optimization  . . . . . . . . . :   *NONE
 Debugging view  . . . . . . . . :   *NONE
 Define names  . . . . . . . . . :
 Language level  . . . . . . . . :   *SOURCE
 Source margins:
   Left margin . . . . . . . . . :   1
   Right margin  . . . . . . . . :   32754
 Sequence columns:
   Left Column . . . . . . . . . :
   Right Column  . . . . . . . . :
 Message flagging level  . . . . :   0
 Compiler messages:
   Message limit . . . . . . . . :   *NOMAX
   Message limit severity  . . . :   30
 Replace module object . . . . . :   *YES
 User Profile  . . . . . . . . . :   *USER
 Authority . . . . . . . . . . . :   *LIBCRTAUT
 Target release  . . . . . . . . :   *CURRENT
 System includes . . . . . . . . :   *YES
 Last change . . . . . . . . . . :   02/11/94 12:33:08
 Source description  . . . . . . :   APPC C program example CPIC - Source
 Compiler  . . . . . . . . . . . :   IBM ILE C/400 Compiler
                                          * * * * *  S O U R C E  * * * * *
 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
    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 (CMINIT)          */
    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 6). ILE 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 inquries 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                        */
   53       |/*         CM_COMPLETE_DATA_RECEIVED, or if the                     */

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
   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 <recio.h>               /* Record i/o routines            */
   75       |#include "QSYSINC/H/CMC"         /* CPI-Communications pseudonyms  */
   76       |
   77       |/******************************************************************/
   78       |/* Define variables used with CPI-Communications calls.             */
   79       |/******************************************************************/
   80       |CM_INT32 data_received;
   81       |CM_INT32 requested_length;
   82       |CM_INT32 received_length;
   83       |CM_INT32 request_to_send_received;
   84       |CM_INT32 return_code;
   85       |CM_INT32 send_length;
   86       |CM_INT32 send_type;
   87       |CM_INT32 status_received;
   88       |CM_INT32 deallocate_type;
   89       |CM_INT32 sync_level;
   90       |CM_INT32 TP_name_length;
```

*Figure F-4 (Part 2 of 6). ILE C/400 Inquiry Example – Local Program*

```
   91       |char    conversation_ID??(8??);
   92       |char    sym_dest_name??(8??);
   93       |char    TP_name??(8??);
   94       |
   95       |/*********************************************************************/
   96       |/* Define the structures used for reads/writes from/to the display   */
   97       |/* file.                                                             */
   98       |/*********************************************************************/
   99       | struct {
  100       |   char partn??(5??);             /* part number                    */
  101       |   char partd??(25??);            /* part description               */
  102       |   char errorl??(40??);           /* error record                   */
  103       |} prompt_i_o = { "     ", "                          ",
  104       |                 "                                   "   };
  105       |
  106       |/*********************************************************************/

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  107       |/* Define constants/flags used in program.                           */
  108       |/*********************************************************************/
  109       |#define ERROR 1                  /* error during IO processing      */
  110       |#define NOERROR 0
  111       |
  112       |/*********************************************************************/
  113       |/* Declare global variables/functions.                               */
  114       |/*********************************************************************/
  115       |void cleanup(int);
  116       |void open_files(void);
  117       |void get_cust_num(void);
  118       |void start_conversation(void);
  119       |
  120       |_RFILE *dspfptr;                  /* Pointer to the display file     */
  121       |_XXIOFB_DSP_ICF_T *dsp_icf_fdbk;  /* IO Feedback - display file      */
  122       |_SYSindara dsp_indic;             /* indicator area for dsp          */
  123       |char blank40??(40??) = "                                        ";
  124       |
  125       |
  126       |/*********************************************************************/
  127       |/* START OF PROGRAM                                                  */
  128       |/*                                                                   */
  129       |/* Files are opened, a conversation with the remote program is       */
  130       |/* started, and the part inquiry screen is displayed.  Inquiries     */
  131       |/* are handled until the user presses the F3=Exit key, in which case */
  132       |/* the conversation will be ended and the program will end.          */
  133       |/*********************************************************************/
  134       |main()
  135       |{
  136     1 | open_files();
  137     2 | start_conversation();
  138       |
  139     3 | get_cust_num();
  140       |
  141     4 | while (dsp_indic??(98??) != '1')
  142       |   {
  143       |
  144       |    /*****************************************************************/
  145       |    /* The part number that the user has requested information       */
  146       |    /* for is sent to the remote program using the CMSEND call.      */
  147       |    /*****************************************************************/
  148     5 |    send_length = 5;
  149       |    CMSEND(conversation_ID, prompt_i_o.partn, &send_length,;
  150     6 |           &request_to_send_received,; &return_code);;
  151       |
```

*Figure F-4 (Part 3 of 6). ILE C/400 Inquiry Example – Local Program*

```
  152       |    /*************************************************************/
  153       |    /* The CMRCV call is issued to receive the response          */
  154       |    /* from the remote program (the response can either be       */
  155       |    /* an error message or the part description, depending       */
  156       |    /* on whether the part was found or not).                    */
  157       |    /*************************************************************/
  158     7 |    if (return_code == CM_OK)
  159       |      {

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  160     8 |       requested_length = 25;
  161       |       CMRCV(conversation_ID, prompt_i_o.partd, &requested_length,;
  162       |             &data_received,; &received_length,; &status_received,;
  163     9 |             &request_to_send_received,; &return_code);;
  164       |
  165    10 |       strncpy(prompt_i_o.errorl, blank40, 40);
  166       |      }
  167       |    else
  168    11 |      if (return_code == CM_PROGRAM_ERROR_PURGING)
  169       |        {
  170    12 |         requested_length = 40;
  171       |         CMRCV(conversation_ID, prompt_i_o.errorl, &requested_length,;
  172       |               &data_received,; &received_length,; &status_received,;
  173    13 |               &request_to_send_received,; &return_code);;
  174       |
  175    14 |         strncpy(prompt_i_o.partd, "                        ", 25);
  176       |        }
  177       |      else                       /* Unexpected return code.         */
  178    15 |        cleanup(ERROR);
  179       |
  180       |    if ((return_code != CM_OK) ||
  181       |        (data_received != CM_COMPLETE_DATA_RECEIVED) ||
  182    16 |        (status_received != CM_SEND_RECEIVED))
  183    17 |      cleanup(ERROR);
  184       |
  185    18 |    get_cust_num();
  186       |   }                             /* end of while                    */
  187       |
  188    19 | cleanup(NOERROR);
  189       |
  190       |}                                /* end of MAIN routine             */
  191       |
  192       |/*************************************************************************/
  193       |/*                                                                       */
  194       |/*   ****************                                ****************   */
  195       |/*   *                  INTERNAL FUNCTIONS                        *   */
  196       |/*   ****************                                ****************   */
  197       |/*                                                                       */
  198       |/*************************************************************************/
  199       |
  200       |/*************************************************************************/
  201       |/* "OPEN_FILES" function                                                 */
  202       |/*                                                                       */
  203       |/* This function opens the display file and sets the indicator area. */
  204       |/*************************************************************************/
  205       |void open_files()
  206       |{
  207       | if ((dspfptr=_Ropen("T8189DSP","ar+ indicators=y riofb=y" ))
  208     1 |       == NULL)
  209       |   {
  210     2 |    printf("Display file failed to open.\n");
  211     3 |    exit(ERROR);
  212       |   }
```

*Figure F-4 (Part 4 of 6). ILE C/400 Inquiry Example – Local Program*

```
 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  213       |
  214     4 | _Rindara(dspfptr, dsp_indic);
  215       |
  216     5 | memset(dsp_indic, '0', 99);      /* Initialize indicator area.      */
  217       |}                                 /* end open_files...               */
  218       |
  219       |/*********************************************************************/
  220       |/* "START_CONVERSATION" function                                     */
  221       |/*                                                                   */
  222       |/* This function establishes a conversation with the remote program, */
  223       |/* and also sets various conversation characteristics.               */
  224       |/*********************************************************************/
  225       |void start_conversation()
  226       |{
  227       | /*******************************************************************/
  228       | /* The CMINIT call is issued to initialize various                 */
  229       | /* conversation characteristics.                                   */
  230       | /*******************************************************************/
  231     1 | strncpy(sym_dest_name, "T8189CSI", 8);
  232     2 | CMINIT(conversation_ID, sym_dest_name, &return_code);;
  233     3 | if (return_code != CM_OK)
  234     4 |   cleanup(ERROR);
  235       |
  236       | /*******************************************************************/
  237       | /* The Set_TP_Name (CMSTPN) call is issued so that the             */
  238       | /* TP_name conversation characteristic is set to the remote program.*/
  239       | /*******************************************************************/
  240     5 | strncpy(TP_name, "T8189CCT", 8);
  241     6 | TP_name_length = strlen(TP_name);
  242     7 | CMSTPN(conversation_ID, TP_name, &TP_name_length,; &return_code);;
  243       |
  244       | /*******************************************************************/
  245       | /* The Set_Sync_Level (CMSSL) call is issued so that the           */
  246       | /* sync_level conversation characteristic is set to CM_CONFIRM.    */
  247       | /*******************************************************************/
  248     8 | sync_level = CM_CONFIRM;
  249     9 | CMSSL(conversation_ID, &sync_level,; &return_code);;
  250       |
  251       | /*******************************************************************/
  252       | /* The CMALLC call is issued so that a conversation can be started  */
  253       | /* using the conversation_ID previously assigned by the CMINIT call.*/
  254       | /*******************************************************************/
  255    10 | CMALLC(conversation_ID, &return_code);;
  256    11 | if (return_code != CM_OK)
  257    12 |   cleanup(ERROR);
  258       |
  259       | /*******************************************************************/
  260       | /* The Set_Send_Type (CMSST) call is issued so that the send_type   */
  261       | /* conversation characteristic is set to                           */
  262       | /* CM_SEND_AND_PREP_TO_RECEIVE.                                    */
  263       | /*******************************************************************/
  264    13 | send_type = CM_SEND_AND_PREP_TO_RECEIVE;
  265    14 | CMSST(conversation_ID, &send_type,; &return_code);;

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  266       |
  267       |}                                 /* end start_conversation...       */
  268       |
  269       |/*********************************************************************/
  270       |/* "GET_CUST_NUM" function                                           */
  271       |/*                                                                   */
  272       |/* Get a customer number from the display.                           */
  273       |/*********************************************************************/
  274       |void get_cust_num()
```

*Figure F-4 (Part 5 of 6). ILE C/400 Inquiry Example – Local Program*

```
  275       |{
  276     1 | _Rformat(dspfptr,"PROMPT    ");
  277     2 | _Rwrite (dspfptr, &prompt_i_o,; sizeof(prompt_i_o));
  278     3 | memset(dsp_indic, '0', 99);
  279     4 | _Rreadn (dspfptr, &prompt_i_o,; sizeof(prompt_i_o), __DFT);
  280       |}                                /* end get_cust_num...             */
  281       |
  282       |/*********************************************************************/
  283       |/* "CLEANUP" function.                                               */
  284       |/*                                                                   */
  285       |/* The following code handles the end-of-program processing.         */
  286       |/* This includes the ending of the conversation with                 */
  287       |/* the remote system (if conversation is active) by                  */
  288       |/* issuing a CMDEAL call with the deallocate_type set to either      */
  289       |/* CM_DEALLOCATE_FLUSH  (for non-error conditions),                  */
  290       |/* or CM_DEALLOCATE_ABEND (for error conditions).                    */
  291       |/*********************************************************************/
  292       |void cleanup(int endtype)
  293       |{
  294     1 | if (endtype == NOERROR)
  295       |   {
  296     2 |    deallocate_type = CM_DEALLOCATE_FLUSH;
  297     3 |    CMSDT(conversation_ID, &deallocate_type,; &return_code);;
  298     4 |    CMDEAL(conversation_ID, &return_code);;
  299       |   }
  300       | else
  301       |   if ((return_code != CM_ALLOCATE_FAILURE_RETRY) &&;
  302       |       (return_code != CM_ALLOCATE_FAILURE_NO_RETRY) &&;
  303       |       (return_code != CM_DEALLOCATED_ABEND) &&;
  304       |       (return_code != CM_DEALLOCATED_NORMAL) &&;
  305       |       (return_code != CM_PRODUCT_SPECIFIC_ERROR) &&;
  306       |       (return_code != CM_RESOURCE_FAILURE_RETRY) &&;
  307     5 |       (return_code != CM_RESOURCE_FAILURE_NO_RETRY))
  308       |     {
  309     6 |      deallocate_type = CM_DEALLOCATE_ABEND;
  310     7 |      CMSDT(conversation_ID, &deallocate_type,; &return_code);;
  311     8 |      CMDEAL(conversation_ID, &return_code);;
  312       |     }
  313       |
  314     9 | _Rclose (dspfptr);
  315       |
  316    10 | exit(endtype);
  317       |}                                /* end cleanup...                  */
  318       |

                                    * * * * *  E N D  O F  S O U R C E  * * * * *
                                         * * * * *  I N C L U D E S  * * * * *
INCNBR  Include Name                    Last change        Actual Include Name
    1   stdio.h                         12/02/93 14:12:18  QCLE/H/STDIO
    2   stdlib.h                        12/02/93 14:12:19  QCLE/H/STDLIB
    3   string.h                        12/02/93 14:12:19  QCLE/H/STRING
    4   stddef.h                        12/02/93 14:12:17  QCLE/H/STDDEF
    5   xxfdbk.h                        12/02/93 14:12:23  QCLE/H/XXFDBK
    6   recio.h                         12/02/93 14:12:15  QCLE/H/RECIO
    7   QSYSINC/H/CMC                   01/12/94 16:57:40  QSYSINC/H/CMC
                                   * * * * *  E N D  O F  I N C L U D E S  * * * * *
                                    * * * * *  M E S S A G E  S U M M A R Y  * * * * *
       Total            Informational(00)          Warning(10)          Error(30)           Severe Error(40
         0                    0                          0                0                       0
                              * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * * * *
Program T8189CCS was created in library LAB on 02/11/94 at 12:39:25.
                                 * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

*Figure F-4 (Part 6 of 6). ILE C/400 Inquiry Example – Local Program*

## ILE C/400 Remote Program for Inquiry Applications (Example 1)

The following explains the structure of the ILE C/400 remote program that handles requests sent by the partner program.

## Program Explanation

The reference numbers in the explanation below correspond to the statement numbers in the program example illustrated in Figure F-5 on page F-11.

**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.

**Statement 63** The preprocessor include directive `#include "QSYSINC/H/CMC"` causes the preprocessor to replace the directive with the contents of the AS/400-supplied CPI Communications pseudonym file CMC.

**Statement 68** This structure defines the database file (T8189DB) structures used in the program. T8189DB is the database file used to read the customer records.

**Statement 100** The internal functions are prototyped so the ILE C/400 compiler knows the type of value returned and the type of parameters passed, if any.

**Statement 125** 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.

**Statement 128** The program loops until there are no more requests to process or until an error occurs in the transaction with the partner program.

**Statement 135** A search of the database file is performed using the part number received from the partner program as the key.

**Statement 142** 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.

**Statement 153** 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.

**Statement 166** 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 `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.

**Statement 172** The `cleanup` function is called to perform end-of-program processing.

**Statement 188** The `open_files` function opens the database file.

**Statement 203** The `start_conversation` function establishes a conversation with the partner program and sets various conversation characteristics.

**Statement 210** The Accept_Conversation (CMACCP) call is issued so that a conversation can be started with the partner program.

**Statement 220** The Set_Send_Type (CMSST) call is issued so that the *send_type* conversation characteristic is set to CM_SEND_AND_PREP_TO_RECEIVE.

**Statement 228** 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.

**Statement 238** 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

**Statement 265** The `cleanup` function performs end-of-program processing.

**Statement 267** 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.

**Statement 280** The database file is closed.

```
                                        * * * * *  P R O L O G  * * * * *
 Program . . . . . :   T8189CCT
   Library . . . . . :     LAB
 Source file . . . . . :   QATTSYSC
   Library . . . . . :     QUSRTOOL
 Source member . . . . . :   T8189CCT
 Text Description  . . . . . :   APPC C program example CPIC - Target
 Output  . . . . . :   *PRINT
 Compiler options  . . . . . :   *NOAGR   *NOEXPMAC   *LOGMSG   *NOSECLVL
                               :   *NOSHOWINC   *SHOWSKP   *NOXREF   *USRINCPATH
 Checkout options  . . . . . :   *NOACCURACY   *NOENUM   *NOEXTERN   *NOGENERAL   *NOGOTO   *NOINIT
                               :   *NOPARM   *NOPORT   *NOPPCHECK   *NOPPTRACE
 Optimization  . . . . . :   *NONE
 Debugging view  . . . . . :   *NONE
 Define names  . . . . . :
 Language level  . . . . . :   *SOURCE
 Source margins:
   Left margin . . . . . :   1
   Right margin  . . . . . :   32754
 Sequence columns:
   Left Column . . . . . :
   Right Column  . . . . . :
 Message flagging level  . . . . :   0
 Compiler messages:
   Message limit . . . . . :   *NOMAX
   Message limit severity  . . . :   30
 Replace module object . . . . . :   *YES
 User Profile  . . . . . :   *USER
 Authority . . . . . :   *LIBCRTAUT
 Target release  . . . . . :   *CURRENT
 System includes . . . . . :   *YES
 Last change . . . . . :   02/11/94 12:33:16
 Source description  . . . . . :   APPC C program example CPIC - Target
 Compiler  . . . . . :   IBM ILE C/400 Compiler
                                        * * * * *  S O U R C E  * * * * *
 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
    1       |/*********************************************************************/
    2       |/* Program name.........: T8189CCT                                   */
    3       |/* Program description..: CPI-Communications remote program          */
    4       |/* Language.............: C/400                                      */
    5       |/*                                                                   */
    6       |/* This program accepts the incoming conversation by                 */
    7       |/* issuing an Accept_Conversation (CMACCP) call.  It then            */
    8       |/* issues a Receive (CMRCV) call to receive the part number          */
    9       |/* from the remote system.  When the CMRCV call completes,           */
   10       |/* the status_received value will be CM_CONFIRM_SEND.  The           */
   11       |/* database file T8189DB   is searched for the received part         */
   12       |/* number.  If the part number is found, the Confirmed               */
   13       |/* (CMCFMD) call is issued, followed by a Send_Data                  */
   14       |/* (CMSEND) call (the data sent is the part description              */
   15       |/* corresponding to the part number retrieved from the               */
   16       |/* database file).  However, if the part number is not               */
   17       |/* found, the Send_Error (CMSERR) call is issued, followed           */
   18       |/* by a CMSEND call (the data sent is a message describing           */
   19       |/* the error).  The CMSEND call sending either the part              */
   20       |/* description or the error message is issued with a                 */
   21       |/* send_type conversation characteristic of                          */
   22       |/* CM_SEND_AND_PREP_TO_RECEIVE and a                                 */
   23       |/* prepare_to_receive_type conversation characteristic of            */
   24       |/* CM_PREP_TO_RECEIVE_FLUSH.  Setting the conversation               */
   25       |/* characteristics to these values results in the flushing           */
   26       |/* of the data, and the changing of the data flow                    */
   27       |/* direction.  The partner program can send more inquiries.          */
   28       |/*                                                                   */
   29       |/* This program will continue to handle inquiries from the           */
   30       |/* partner program until a return_code that is not CM_OK             */
   31       |/* is received.  Then the program ends.                              */
```

*Figure F-5 (Part 1 of 6). ILE 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       |/********************************************************************/
   53       |

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
   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 <recio.h>               /* record i/o routines             */
   63       |#include "QSYSINC/H/CMC"         /* CPI-Communications pseudonyms   */
   64       |
   65       |/********************************************************************/
   66       |/* Define the structure used for reads from the database file.      */
   67       |/********************************************************************/
   68       | struct          {
   69       |   char partn??(5??);
   70       |   char partd??(25??);
   71       |} part_rec;
   72       |
   73       |/********************************************************************/
   74       |/* Define variables used with CPI-Communications calls.             */
   75       |/********************************************************************/
   76       |CM_INT32 data_received;
   77       |CM_INT32 prep_to_receive_type;
   78       |CM_INT32 requested_length;
   79       |CM_INT32 received_length;
   80       |CM_INT32 request_to_send_received;
   81       |CM_INT32 return_code;
   82       |CM_INT32 send_length;
   83       |CM_INT32 send_type;
   84       |CM_INT32 status_received;
   85       |CM_INT32 deallocate_type;
   86       |char    conversation_ID??(8??);
   87       |char    buffer??(40??);
   88       |
   89       |/********************************************************************/
   90       |/* Define constants/flags used in program.                          */
   91       |/********************************************************************/
   92       |#define ERROR 1                  /* error during I/O processing     */
   93       |#define NOERROR 0
   94       |#define MORE_REQUESTS 0          /* More request indicator          */
   95       |#define NO_REQUESTS 1
```

*Figure F-5 (Part 2 of 6). ILE C/400 Inquiry Example – Remote Program*

```
  96       |
  97       |/*********************************************************************/
  98       |/* Declare global variables/functions.                               */
  99       |/*********************************************************************/
 100       |int get_cust_num(void);
 101       |void cleanup(void);
 102       |void open_files(void);
 103       |void start_conversation(void);
 104       |
 105       |char part_not_found??(40??) =
 106       |                       "THE REQUESTED PART WAS NOT FOUND       ";

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
 107       |
 108       |_RFILE *dbfptr;                 /* Pointer to database file.        */
 109       |_RIOFB_T *db_fdbk;              /* IO Feedback - data base file     */
 110       |
 111       |size_t           size;          /* "size_t" is a synonym for the    */
 112       |                                /* type of the value returned by    */
 113       |                                /* the "sizeof" operator.           */
 114       |
 115       |
 116       |/*********************************************************************/
 117       |/* START OF PROGRAM                                                  */
 118       |/*                                                                   */
 119       |/* Files are opened, a conversation with the remote program is       */
 120       |/* started, and the part inquiry processing starts.  Inquiries       */
 121       |/* are handled until a CM_DEALLOCATED_NORMAL is received.            */
 122       |/*********************************************************************/
 123       |main()
 124       |{
 125     1 | open_files();
 126     2 | start_conversation();
 127       |
 128     3 | while (get_cust_num() != NO_REQUESTS)
 129       |   {
 130       |    /*****************************************************************/
 131       |    /* A search of the database file is done using the part          */
 132       |    /* number as the key.                                            */
 133       |    /*****************************************************************/
 134     4 |    strncpy (part_rec.partd,"                    ",25);
 135       |    db_fdbk = _Rreadk(dbfptr, &part_rec,; sizeof(part_rec),
 136     5 |              __KEY_EQ, &part_rec;partn, sizeof(part_rec.partn));
 137       |
 138       |    /*****************************************************************/
 139       |    /* If the part number is found, the CMCFMD call is               */
 140       |    /* issued; otherwise, the CMSERR call is issued.                 */
 141       |    /*****************************************************************/
 142     6 |    if (db_fdbk -> num_bytes > 0)    /* if record was found          */
 143       |      {
 144     7 |       CMCFMD(conversation_ID, &return_code);;
 145     8 |       if (return_code != CM_OK)
 146     9 |         cleanup();
 147       |
 148    10 |       strncpy(buffer,part_rec.partd,25);
 149    11 |       send_length = 25;
 150       |      }
 151       |    else                            /* part description not found   */
 152       |      {
 153    12 |       CMSERR(conversation_ID, &request_to_send_received,; &return_code);;
 154    13 |       if (return_code != CM_OK)
 155    14 |         cleanup();
 156       |
 157    15 |       strncpy(buffer, part_not_found, 40);
 158    16 |       send_length = 40;
 159       |      }
```

*Figure F-5 (Part 3 of 6). ILE C/400 Inquiry Example – Remote Program*

```
 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  160       |
  161       |    /*****************************************************************/
  162       |    /* The CMSEND call is issued.  The data sent (set previously)    */
  163       |    /* is either an error message (if the part was not found)        */
  164       |    /* or the part description (if the part is found).               */
  165       |    /*****************************************************************/
  166       |    CMSEND(conversation_ID, buffer, &send_length,;
  167    17 |           &request_to_send_received,; &return_code);;
  168    18 |    if (return_code != CM_OK)
  169    19 |      cleanup();
  170       |   }                              /* end WHILE                     */
  171       |
  172    20 | cleanup();
  173       |}                                 /* end of main routine           */
  174       |
  175       |/*********************************************************************/
  176       |/*                                                                   */
  177       |/*   ****************                                ***************   */
  178       |/*   *                   INTERNAL FUNCTIONS                      *   */
  179       |/*   ****************                                ***************   */
  180       |/*                                                                   */
  181       |/*********************************************************************/
  182       |
  183       |/*********************************************************************/
  184       |/* "OPEN_FILES" function                                             */
  185       |/*                                                                   */
  186       |/* This function opens the database file.                            */
  187       |/*********************************************************************/
  188       |void open_files()
  189       |{
  190     1 | if ((dbfptr= _Ropen("T8189DB", "rr riofb=n")) == NULL)
  191       |   {
  192     2 |    printf("Data Base file failed to open.\n");
  193     3 |    exit(ERROR);
  194       |   }
  195       |}                                 /* end open_files...             */
  196       |
  197       |/*********************************************************************/
  198       |/* "START_CONVERSATION" function                                     */
  199       |/*                                                                   */
  200       |/* This function establishes a conversation with the remote system,  */
  201       |/* and also sets various conversation characteristics.               */
  202       |/*********************************************************************/
  203       |void start_conversation()
  204       |{
  205       |
  206       | /*********************************************************************/
  207       | /* The CMACCP call is issued so that a conversation can be          */
  208       | /* started with the partner program.                                */
  209       | /*********************************************************************/
  210     1 | CMACCP(conversation_ID, &return_code);;
  211     2 | if (return_code != CM_OK)
  212     3 |   cleanup();

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  213       |
  214       | /*********************************************************************/
  215       | /* The Set_Send_Type (CMSST) call is issued so that the             */
  216       | /* send_type conversation characteristic is set to                  */
  217       | /* CM_SEND_AND_PREP_TO_RECEIVE.                                     */
  218       | /*********************************************************************/
```

*Figure F-5 (Part 4 of 6). ILE C/400 Inquiry Example – Remote Program*

```
 219     4 | send_type = CM_SEND_AND_PREP_TO_RECEIVE;
 220     5 | CMSST(conversation_ID, &send_type,; &return_code);;
 221       |
 222       | /*********************************************************************/
 223       | /* The Set_Prepare_To_Receive_Type (CMSPTR) call is issued          */
 224       | /* so that the prepare_to_receive_type conversation                 */
 225       | /* characteristic is set to CM_PREP_TO_RECEIVE_FLUSH.               */
 226       | /*********************************************************************/
 227     6 | prep_to_receive_type = CM_PREP_TO_RECEIVE_FLUSH;
 228     7 | CMSPTR(conversation_ID, &prep_to_receive_type,; &return_code);;
 229       |
 230       |}                                /* end start_conversation...       */
 231       |
 232       |/*********************************************************************/
 233       |/* "GET_CUST_NUM" function                                           */
 234       |/*                                                                   */
 235       |/* This subroutine waits for incoming data from the partner          */
 236       |/* program by issuing the read operation.                            */
 237       |/*********************************************************************/
 238       |get_cust_num()
 239       |{
 240     1 | requested_length = 5;
 241       | CMRCV(conversation_ID, part_rec.partn, &requested_length,;
 242       |       &data_received,; &received_length,; &status_received,;
 243     2 |       &request_to_send_received,; &return_code);;
 244       |
 245     3 | if (return_code == CM_OK)
 246       |   {
 247       |    if ((data_received == CM_COMPLETE_DATA_RECEIVED) &&;
 248     4 |        (status_received == CM_CONFIRM_SEND_RECEIVED))
 249     5 |      return(MORE_REQUESTS);
 250       |    else
 251     6 |      return(NO_REQUESTS);
 252       |   }
 253       | else
 254     7 |   return(NO_REQUESTS);
 255       |}                                /* end get_cust_num...             */
 256       |
 257       |/*********************************************************************/
 258       |/* "CLEANUP" function.                                               */
 259       |/*                                                                   */
 260       |/* The following code handles the end-of-program processing.         */
 261       |/* This includes the ending of the conversation with                 */
 262       |/* the remote system (if conversation is active), and the            */
 263       |/* closing of opened files.                                          */
 264       |/*********************************************************************/
 265       |void cleanup()

 Line  STMT
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
 266       |{
 267       | if ((return_code != CM_ALLOCATE_FAILURE_RETRY) &&;
 268       |     (return_code != CM_ALLOCATE_FAILURE_NO_RETRY) &&;
 269       |     (return_code != CM_DEALLOCATED_ABEND) &&;
 270       |     (return_code != CM_DEALLOCATED_NORMAL) &&;
 271       |     (return_code != CM_PRODUCT_SPECIFIC_ERROR) &&;
 272       |     (return_code != CM_RESOURCE_FAILURE_RETRY) &&;
 273     1 |     (return_code != CM_RESOURCE_FAILURE_NO_RETRY))
 274       |   {
 275     2 |    deallocate_type = CM_DEALLOCATE_ABEND;
 276     3 |    CMSDT(conversation_ID, &deallocate_type,; &return_code);;
 277     4 |    CMDEAL(conversation_ID, &return_code);;
 278       |   }
 279       |
```

*Figure F-5 (Part 5 of 6). ILE C/400 Inquiry Example – Remote Program*

```
280     5 | _Rclose(dbfptr);
  281       |
  282     6 | exit(0);
  283       |}                                  /* end cleanup...                */
  284       |
                                          * * * * *  E N D  O F  S O U R C E  * * * * *
                                              * * * * *  I N C L U D E S  * * * * *
INCNBR  Include Name                      Last change       Actual Include Name
    1   stdio.h                           12/02/93 14:12:18  QCLE/H/STDIO
    2   stdlib.h                          12/02/93 14:12:19  QCLE/H/STDLIB
    3   string.h                          12/02/93 14:12:19  QCLE/H/STRING
    4   stddef.h                          12/02/93 14:12:17  QCLE/H/STDDEF
    5   xxfdbk.h                          12/02/93 14:12:23  QCLE/H/XXFDBK
    6   recio.h                           12/02/93 14:12:15  QCLE/H/RECIO
    7   QSYSINC/H/CMC                     01/12/94 16:57:40  QSYSINC/H/CMC
                                        * * * * *  E N D  O F  I N C L U D E S  * * * * *
                                         * * * * *  M E S S A G E  S U M M A R Y  * * * * *
        Total           Informational(00)          Warning(10)          Error(30)           Severe Error(40
          0                     0                       0                    0                      0
                                * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * * * *
Program T8189CCT was created in library LAB on 02/11/94 at 12:40:51.
                                   * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

*Figure F-5 (Part 6 of 6). ILE 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-18.

**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, “ ICF Program Examples” 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  . . . . . :     QUSRTOOL
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 . . . . . :   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.  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*************************************************************
  144  001820     MOVE "T8189CSI" TO SYM-DEST-NAME.
 9 145 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 (CMSSL) call is issued so that the     *
       002050* sync_level conversation characteristic is set to          *
       002060* CM_CONFIRM.                                               *
       002070*************************************************************
       002080
  152  002090     SET CM-CONFIRM TO TRUE.
11 153 002100     CALL "CMSSL" 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*************************************************************
       002410
14     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-27.

**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 (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 `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 ITEMNM.
       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).                                  <-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
   21 +000005       05  DBRCD         REDEFINES T8189DB-RECORD.                  <-ALL-FMTS
   22 +000006           06 ITEMNM                PIC X(5).                        <-ALL-FMTS
   23 +000007           06 ITEMD                 PIC X(25).                       <-ALL-FMTS
       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                                                        CMCOBOL
      +000020*************************************************                  CMCOBOL
      +000030* NOTE: BUFFER MUST BE DEFINED IN WORKING STORAGE *                  CMCOBOL
      +000040*************************************************                  CMCOBOL
      +000050*                                                                  CMCOBOL
   30 +000060 01  CONVERSATION-ID            PIC X(8).                          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*                                                                CMCOBOL
 83 +000750 01  PARTNER-LU-NAME            PIC X(17).                       CMCOBOL
    +000760*    1-17 BYTES                                                  CMCOBOL
    +000770*                                                                CMCOBOL
 84 +000780 01  PARTNER-LU-NAME-LENGTH     PIC 9(9) COMP-4.                 CMCOBOL
    +000790*                                                                CMCOBOL
 85 +000800 01  PREPARE-TO-RECEIVE-TYPE    PIC 9(9) COMP-4.                 CMCOBOL
 86 +000810     88   CM-PREP-TO-RECEIVE-SYNC-LEVEL  VALUE 0.               CMCOBOL
 87 +000820     88   CM-PREP-TO-RECEIVE-FLUSH       VALUE 1.               CMCOBOL
 88 +000830     88   CM-PREP-TO-RECEIVE-CONFIRM     VALUE 2.               CMCOBOL
    +000840*                                                                CMCOBOL
 89 +000850 01  RECEIVED-LENGTH            PIC 9(9) COMP-4.                 CMCOBOL
    +000860*                                                                CMCOBOL
 90 +000870 01  RECEIVE-TYPE               PIC 9(9) COMP-4.                 CMCOBOL
 91 +000880     88   CM-RECEIVE-AND-WAIT   VALUE 0.                         CMCOBOL
 92 +000890     88   CM-RECEIVE-IMMEDIATE  VALUE 1.                         CMCOBOL
    +000900*                                                                CMCOBOL
 93 +000910 01  REQUESTED-LENGTH           PIC 9(9) COMP-4.                 CMCOBOL
    +000920*                                                                CMCOBOL
 94 +000930 01  REQUEST-TO-SEND-RECEIVED   PIC 9(9) COMP-4.                 CMCOBOL
 95 +000940     88  CM-REQ-TO-SEND-NOT-RECEIVED    VALUE 0.                CMCOBOL
 96 +000950     88  CM-REQ-TO-SEND-RECEIVED        VALUE 1.                CMCOBOL
    +000960*                                                                CMCOBOL
 97 +000970 01  RETURN-CONTROL             PIC 9(9) COMP-4.                 CMCOBOL
 98 +000980     88   CM-WHEN-SESSION-ALLOCATED      VALUE 0.               CMCOBOL
 99 +000990     88   CM-IMMEDIATE                   VALUE 1.               CMCOBOL
    +001000*                                                                CMCOBOL
100 +001010 01  SEND-LENGTH                PIC 9(9) COMP-4.                 CMCOBOL
    +001020*                                                                CMCOBOL
101 +001030 01  SEND-TYPE                  PIC 9(9) COMP-4.                 CMCOBOL
102 +001040     88   CM-BUFFER-DATA                 VALUE 0.               CMCOBOL
103 +001050     88   CM-SEND-AND-FLUSH              VALUE 1.               CMCOBOL
104 +001060     88   CM-SEND-AND-CONFIRM            VALUE 2.               CMCOBOL
105 +001070     88   CM-SEND-AND-PREP-TO-RECEIVE    VALUE 3.               CMCOBOL
106 +001080     88   CM-SEND-AND-DEALLOCATE         VALUE 4.               CMCOBOL
    +001090*                                                                CMCOBOL
107 +001100 01  STATUS-RECEIVED            PIC 9(9) COMP-4.                 CMCOBOL
108 +001110     88  CM-NO-STATUS-RECEIVED        VALUE 0.                  CMCOBOL
109 +001120     88  CM-SEND-RECEIVED             VALUE 1.                  CMCOBOL
110 +001130     88  CM-CONFIRM-RECEIVED          VALUE 2.                  CMCOBOL
111 +001140     88  CM-CONFIRM-SEND-RECEIVED     VALUE 3.                  CMCOBOL
112 +001150     88  CM-CONFIRM-DEALLOC-RECEIVED  VALUE 4.                  CMCOBOL
    +001160*                                                                CMCOBOL
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.                 CMCOBOL
114 +001180     88   CM-NONE               VALUE 0.                         CMCOBOL
115 +001190     88   CM-CONFIRM            VALUE 1.                         CMCOBOL
    +001200*                                                                CMCOBOL
116 +001210 01  SYM-DEST-NAME              PIC X(8).                        CMCOBOL
    +001220*                                                                CMCOBOL
117 +001230 01  TP-NAME                    PIC X(64).                       CMCOBOL
    +001240*    1-64 BYTES                                                  CMCOBOL
    +001250*                                                                CMCOBOL
118 +001260 01  TP-NAME-LENGTH             PIC 9(9) COMP-4.                 CMCOBOL
     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      001290 OPEN-FILES.
       001300
  125  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      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     001810 HANDLE-INQUIRY.
  133  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 CMCFMD 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 ITEMD 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                        ITEMNM
       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-35.

**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 QRPG/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, “ ICF Program Examples” 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 QRPGMSG.
   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                                                                  IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
                     S o u r c e   L i s t i n g
      10  H
 1    20  FT8189DSPCF  E                    WORKSTN
      30  F                                           KINFSR FAIL
      40  I*************************************************************
      50  I* Program name.........: T8189CRS                          *
      60  I* Program description..: CPI Communications local program   *
      70  I* Language.............: RPG/400                           *
      80  I*                                                          *
      90  I* This program invokes a program to handle part inquiry on  *
     100  I* the remote system.  The Initialize_Conversation (CMINIT)  *
     110  I* call is issued using the sym_dest_name of 'T8189CSI'.     *
     120  I* The Allocate (CMALLC) call is issued, which results in    *
     130  I* the establishment of a conversation with the remote       *
     140  I* program.  A display which prompts the user for the part   *
     150  I* number for which part information is requested is then    *
     160  I* displayed.  When the user presses Enter, a Send_Data      *
     170  I* (CMSEND) call is issued (the data sent to the partner     *
     180  I* program is the part number).  Note that the CMSEND call   *
     190  I* is issued with the following conversation                 *
     200  I* characteristics: a send_type of                          *
     210  I* CM_SEND_AND_PREP_TO_RECEIVE; a prepare_to_receive_type    *
     220  I* of CM_PREP_TO_RECEIVE_SYNC_LEVEL; and a sync_level of     *
     230  I* CM_CONFIRM.  Setting the conversation characteristics to  *
     240  I* these values results in the flushing of the data, the     *
     250  I* changing of the data flow direction, and the sending of   *
     260  I* a confirmation request to the partner program.  If the    *
     270  I* partner program responds with the Confirmed (CMCFMD)      *
     280  I* call to the confirmation request, the return_code         *
     290  I* parameter value on the CMSEND call will be set to CM_OK;  *
     300  I* the Receive (CMRCV) call is then issued to receive the    *
     310  I* part description.  However, if the partner program        *
     320  I* responds with the Send_Error (CMSERR) call to the         *
     330  I* confirmation request, the return_code parameter value on  *
     340  I* the CMSEND call will be set to CM_PROGRAM_ERROR_PURGING;  *
     350  I* a CMRCV call is issued to receive the error message.      *
     360  I*                                                          *
     370  I* The error message or part description (depending on       *
     380  I* whether the part number was found) will be displayed on   *
     390  I* the screen.                                              *
     400  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 QRPG/QIRGINC,CMRPG
 A000000+  MEMBER CMRPG IN FILE QIRGINC LIBRARY QRPG 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         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
SEQUENCE                                                                IND   DO   LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  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   (SNDCNF)
A001530+ I*   CM_SEND_AND_PREP_TO_RECEIVE    -- VALUE 3   (SNDPTR)
A001540+ I*   CM_SEND_AND_DEALLOCATE         -- VALUE 4   (SNDDEL)
```

*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         SNDDEL
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                                                                IND   DO    LAST       PAGE   PROGRAM
 NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE   NUM   UPDATE     LINE   ID
    750  ICMPARM      DS
    760  I                                        1   8 CONVID
    770  I                                    B   9  12 0RTNCOD
    780  I                                    B  13  16 0DATRCV
    790  I                                    B  17  20 0DLCTYP
    800  I                                    B  21  24 0RCVLEN
    810  I                                    B  25  28 0REQLEN
    820  I                                    B  29  32 0REQTSR
    830  I                                    B  33  36 0SNDLEN
    840  I                                    B  37  40 0SNDTYP
    850  I                                    B  41  44 0STSRCV
    860  I                                    B  45  48 0SYNLVL
    870  I                                       49  56  SYMDST
    880  I                                       57  64  TPNAM
    890  I                                    B  65  68 0TPNLEN
    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                                                              IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
    1270  C                   CALL 'CMRCV'                                    002
    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'                                    003
    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                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
    1810  C           RTNCOD    ANDNERFRET                                      001
    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           RTNCOD                           001
    1900  C                     CALL 'CMDEAL'                                   001
    1910  C                     PARM           CONVID                           001
    1920  C                     PARM           RTNCOD                           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                   MOVEL'T8189CSI'SYMDST  8
 12 2210  CSR                   CALL 'CMINIT'
    2220  CSR                   PARM           CONVID
    2230  CSR                   PARM           SYMDST
    2240  CSR                   PARM           RTNCOD
    2250  CSR         RTNCOD    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                   MOVEL'T8189CRT'TPNAM   8
    2340  CSR                   Z-ADD8         TPNLEN
SEQUENCE                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
 13 2350  CSR                   CALL 'CMSTPN'
    2360  CSR                   PARM           CONVID
    2370  CSR                   PARM           TPNAM
    2380  CSR                   PARM           TPNLEN
    2390  CSR                   PARM           RTNCOD
    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*
    2470  CSR                   Z-ADDCONFRM    SYNLVL
14  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    CABNECMOK      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                                                                    IND    DO    LAST       PAGE   PROGRAM
 NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
  C000002                         PARTD     30  CHAR   25
  C000003                         ERRORL    70  CHAR   40
          * * * * *  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.
* 7089       20   RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE T8189DSP.
                      C r o s s  R e f e r e n c e
 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  CMCONS      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  CNVMIS      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-46.

**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 QRPG/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 QRPGMSG.
   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 . . . . . :   T8189CRT
   File . . . . . :   QATTRPG
   Library  . . . . . :   QUSRTOOL
   Last Change  . . . . . :   09/26/90  08:27:42
SEQUENCE                                                                  IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
                      S o u r c e   L i s t i n g
      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                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   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 QRPG/QIRGINC,CMRPG
  A000000+  MEMBER CMRPG IN FILE QIRGINC LIBRARY QRPG 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 ITEMNM
  B000002+                                      6  30 ITEMD
  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                                                                IND   DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE   NUM   UPDATE     LINE   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                                                                IND   DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE   NUM   UPDATE     LINE   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_PGM   -- 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  (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         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         DABSVC
A001350+ I              31                    C         DABTIM
A001360+ I              32                    C         SVCENT
A001370+ I              33                    C         SVCEP
A001380+ I              34                    C         SVCET
A001390+ I*****************************************************************
SEQUENCE                                                          IND   DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  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   (SNDDEL)
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         SNDDEL
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  120RTNCOD
      670  I                                   B  13  160DATRCV
      680  I                                   B  17  200DLCTYP
      690  I                                   B  21  240RCVLEN
      700  I                                   B  25  280REQLEN
      710  I                                   B  29  320REQTSR
 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  480PRERCV
      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           RTNCOD    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               MOVELMSG,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...*  USE    NUM   UPDATE     LINE   ID
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*
    1900  CSR                   Z-ADDPTRFLS    PRERCV
13  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           ITEMNM
    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                                      B001
    2310  CSR         DATRCV    IFNE COMDAT                                    B002
SEQUENCE                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
    2320  CSR         STSRCV    ORNE CONSND                                     002
    2330  CSR                   GOTO ENDCNV                                     002
    2340  CSR                   END                                            E002
    2350  CSR                   END                                            E001
    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                                                                                                    LAST
 NUMBER    *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8               UPDATE
                   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
                     PHYSICAL    LOGICAL
        FILE/RCD     FIELD       FIELD       ATTRIBUTES
    01  T8189DB
            DBRCD
                     ITEMNM                  CHAR    5
                       C r o s s   R e f e r e n c e
 File and Record References:
        FILE/RCD    DEV/RCD     REFERENCES (D=DEFINED)
    01  T8189DB     DISK             20D
          DBRCD                      20D B000000      970
 Field References:
        FIELD       ATTR    REFERENCES (M=MODIFIED D=DEFINED)
        ALFLNR      CONST   A001120D    1440
        ALFLRE      CONST   A001130D    1450
* 7031  BASIC       CONST   A000110D
* 7031  BUFDAT      CONST   A001560D
* 7031  CMCONS      DS(1)   A000040D
        CMOK        CONST   A001110D     890     1160     1300     1820
                                2300
* 7031  CMPARM      DS(88)      640D
* 7031  CNVMIS      CONST   A001140D
        COMDAT      CONST   A000230D    2310
* 7031  CONDEL      CONST   A001740D
* 7031  CONFRM      CONST   A001820D
* 7031  CONRCV      CONST   A001720D
        CONSND      CONST   A001730D    2320
        CONVID      A(8)         650D    1070     1140     1250     1530
                                1570    1800     1920     2040     2210
* 7031  DABSVC      CONST   A001340D
* 7031  DABTIM      CONST   A001350D
        DATRCV      B(9,0)       670D    2240     2310
* 7031  DATREC      CONST   A000220D
        DEABND      CONST   A001220D    1460
        DEABTY      CONST   A000360D    1510
* 7031  DECONF      CONST   A000350D
* 7031  DEFLUS      CONST   A000340D
        DENORM      CONST   A001230D    1470
* 7031  DESYNC      CONST   A000330D
        DLCTYP      B(9,0)       680D    1510M    1540
        ENDCNV      TAG         1160     1300     1420D    2330     2450
        ENDPGM      TAG         1610D    1820
        FAIL        BEGSR         20     2440D
* 7031  FILBUF      CONST   A000520D
* 7031  FILLL       CONST   A000510D
        GETDTA      BEGSR        880     1320     2180D
* 7031  IMMED       CONST   A001460D
* 7031  INCDAT      CONST   A000240D
        ITEMD       A(25)   B000002D    1120
        ITEMNM      A(5)    B000001D     970     2220
* 7031  MAPPED      CONST   A000120D
```

*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
        PRERCV      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
        STSRCV      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*

# Appendix G. APPC Tools

## AS/400 APPC File Transfer Protocol

The APPC File Transfer Protocol (AFTP) is an advanced program-to-program communications (APPC) application protocol that provides file transfer services to application programs and end-users. With AFTP, you can copy text and binary files between your computer and any computer that is running the AFTP server.

Contact your IBM market representative or IBM business partner for additional information about AS/400 APPC File Transfer Protocol.

## ATELL Tools

ATELL is an APPC tool that is available in the library QUSRTOOL. Information about his tool is available in the file QUSRTOOL/QATTINFO. The member name is:

- TLTINFO (Information on ATELL)

## ATELL

ATELL is a sample program that allows a workstation user to send a message to another workstation. ATELL is written in the C language and uses CPI Communications.

ATELL is made up of two transaction programs: ATELL, which runs on the client side, and ATELLD, which runs on the server side.

## Installing the tool ATELL

In order to use the ATELL tool, you must first create it. Building the objects involves compiling the ILE C modules used by ATELL and creating the ATELL and ATELLD programs. This is done by creating a CL program that builds the objects you need. Table G-1 lists the source you need to create the ATELL tool.

*Table G-1. Source for the ATELL Tool*

| File | Member | Description. |
|---|---|---|
| QATTINFO | TLTINFO | Information on how to create, use and delete the ATELL tool. |
| QATTCL | TLTCRT | The CL program source for the installation of ATELL. |
| QATTCL | TLTDLT | The CL program source for deleting the tool ATELL. |
| QATTSYSC | TLTATELL | C program source for the module ATELL. |
| QATTSYSC | TLTATELD | C program source for the module ATELLD. |
| QATTSYSC | TLTCERR | C program source for the module CPICERR. |
| QATTSYSC | TLTCINIT | C program source for the module CPICINIT. |
| QATTSYSC | TLTCPORT | C program source for the module CPICPORT. |
| QATTSYSC | TLTCGOPT | C program source for the module GETOPT. |
| QATTSYSC | TLTHCPIC | C header file. |
| QATTSYSC | TLTHCMC | C header file. |
| QATTSYSC | TLTHERR | C header file. |
| QATTSYSC | TLTHDEFS | C header file. |
| QATTSYSC | TLTHPORT | C header file. |
| QATTSYSC | TLTHINIT | C header file. |
| QATTSYSC | TLTHGOPT | C header file. |

To create the ATELL tool you will need to do the following:

1. Create the CL installation program TLTCRT in library MYLIB by entering:

   ```
   CRTCLPGM PGM(MYLIB/TLTCRT) SRCFILE(QUSRTOOL/QATTCL)
   ```

   Where MYLIB is the library in which you want the CL program to exist.

2. To call the installation program, enter:

   ```
   CALL MYLIB/TLTCRT ATELL
   ```

   Where ATELL is the library in which you want the tools ATELL and ATELLD to exist. If this library does not already exist it will be created.

Table G-2 on page G-2 lists the objects created in the library ATELL.

*Table G-2. Objects Created for the ATELL Tool*

| Object Name | Object Type | Description. |
|---|---|---|
| ATELL | *PGM | The ATELL program. |
| ATELLD | *PGM | The ATELLD program. |
| ATELL | *MODULE | The ATELL module used in the ATELL program. |
| ATELLD | *MODULE | The ATELLD module used in the ATELLD program. |
| CPICERR | *MODULE | The CPICERR module used in both ATELL and ATELLD. |
| CPICINIT | *MODULE | The CPICINIT module used in both ATELL and ATELLD. |
| CPICPORT | *MODULE | The CPICPORT module used in both ATELL and ATELLD. |
| GETOPT | *MODULE | The GETOPT module used in both ATELL and ATELLD. |

## Deleting the tool ATELL

To delete the ATELL tool create the CL delete program TLTDLT in library MYLIB:

```
CRTCLPGM PGM(MYLIB/TLTDLT) SRCFILE(QUSRTOOL/QATTCL)
```

Where MYLIB is the library in which you want the CL program to exist.

Once the delete program is created you can do one of the following:

1. If you want to delete only the source members in QUSRTOOL, enter:

   ```
   CALL MYLIB/TLTDLT (*YES *NONE)
   ```

2. If you want to delete only the library ATELL, enter:

   ```
   CALL MYLIB/TLTDLT (*NO ATELL)
   ```

   Where ATELL is the library in which you created the tools ATELL and ATELLD.

3. If you want to delete both the source members in QUSRTOOL and the library ATELL, enter:

   ```
   CALL MYLIB/TLTDLT (*YES ATELL)
   ```

   Where ATELL is the library in which you created the tools ATELL and ATELLD.

## Configuration Requirements for Using ATELL

Before using ATELL there are configuration changes that will need to be done. Refer to the information in the file QATTINFO, member TLTINFO for details on these changes.

**Note:** ATELL must be installed on both the client and the server system.

## Calling ATELL

To see the parameters used in calling ATELL, enter:

```
CALL ATELL/ATELL
```

An example call follows:

```
CALL ATELL/ATELL (USER001@NETSNA.CLIENT 'TESTING TESTING')
```

For more information on ATELL parameters, refer to the information file QATTINFO, member TLTINFO.

## Installation Example of an APPC Tool

The following screens show an example installation of the tool ATELL.

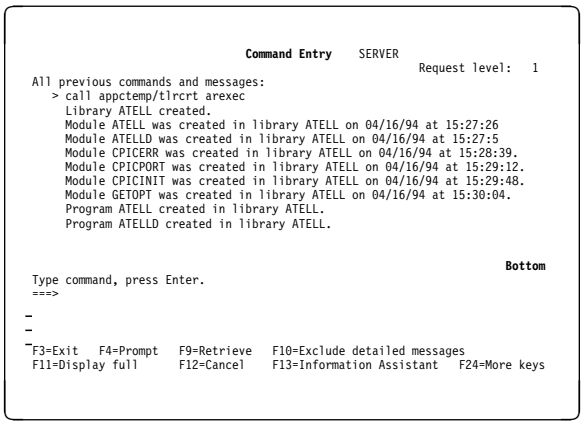
After creating the library in which you want the CL installation program to exist, create the CL program by entering the following:

```
                              Command Entry    SERVER
                                                      Request level:   1
 Previous commands and messages:

    > CRTLIB LIB(MYLIB) TEXT('Library for APPC tools')
      Library MYLIB created.









                                                                     Bottom
 Type command, press Enter.
  ===> CRTCLPGM PGM(MYLIB/TLRCRT) SRCFILE(QUSRTOOL/QATTCL)
TEXT('Program for
creating ATELL')
-
-
 F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
 F11=Display full      F12=Cancel    F13=Information Assistant   F24=More keys
```

Now, call the installation program passing the library for the tools to reside in as a parameter.

```
                              Command Entry    SERVER
                                                      Request level:   1
 Previous commands and messages:

    > CRTLIB LIB(MYLIB) TEXT('Library for APPC tools')
      Library MYLIB created.

    > CRTCLPGM PGM(MYLIB/TLRCRT) SRCFILE(QUSRTOOL/QATTCL) TEXT('Program for
      creating ATELL')









                                                                     Bottom
 Type command, press Enter.
  ===> call mylib/tlrcrt atell
-
-
-
 F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
 F11=Display full      F12=Cancel    F13=Information Assistant   F24=More keys
```

The following screen shows the resulting objects that are created as a result of running the installation program. (Note that PF10 has been used to include the detailed messages generated by the call to TLRCRT.)



# Bibliography

This section lists a subset of IBM publications that contain information about topics described or referred to in this guide.

## Planning and Installation Books

The following planning and installation book contains information you may need when you use the AS/400 APPC support.

- *Local Device Configuration*, SC41-5121

## Customer and System Operation Books

The following customer and system operation books contain information you may need when you use the AS/400 APPC support.

- *Work Management*, SC41-5306
- *Backup and Recovery*, SC41-5304
- *System Operation*, SC41-4203
- *Basic System Operation, Administration, and Problem Handling*, SC41-5206

## AS/400 Communications Books

The following AS/400 communications books contain information you may need when you use the AS/400 APPC support.

- *APPN Support*, SC41-5407
- *CICS/400 Administration and Operations Guide*, SC33-1387
- *CICS/400 Application Programming Guide*, SC33-1386
- *SNA Distribution Services*, SC41-5410
- *ISDN Support*, SC41-5403
- *ICF Programming*, SC41-5442
- *Sockets Programming*, SC41-5422
- *Communications Configuration*, SC41-5401
- *DSNX Support*, SC41-5409
- *Remote Work Station Support*, SC41-5402
- *TCP/IP Configuration and Reference*, SC41-5420
- *3270 Device Emulation Support*, SC41-5408

## AS/400 Programming Books

The following AS/400 programming books contain information you may need when you use the AS/400 APPC support.

- *DDS Reference*, SC41-5712
- *Distributed Data Management*, SC41-5307
- *CL Programming*, SC41-5721
- *CL Reference*, SC41-5722
- *Security - Reference*, SC41-5302

## Client Access/400 Books

- *Client Access/400 for DOS with Extended Memory Setup*, SC41-3500
- *Client Access/400 for DOS with Extended Memory User Guide*, SC41-3501
- *Client Access/400 for DOS with Extended Memory Setup (DBCS)*, SC41-3502
- *Client Access/400 for DOS with Extended Memory User Guide (DBCS)*, SC41-3503
- *Client Access/400 for OS/2 Setup*, SC41-3520
- *Client Access/400 for OS/2 User Guide*, SC41-3521
- *Client Access/400 for OS/2 Setup (DBCS)*, SC41-3522
- *Client Access/400 for OS/2 User Guide (DBCS)*, SC41-3523
- *Client Access/400 for DOS and OS/2 API Reference*, SC41-3562
- *Client Access/400 for DOS and OS/2 Technical Reference*, SC41-3563

## Communications Architectures

The following manuals provide information on the communications protocols: SNA 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.

The following CD-ROM provides information about APPC, APPN, and CPI Communications.

- *The Best of APPC, APPN and CPI-C Collection Kit*, SK2T-2013.

## CPI Communications

- *CPI Communications Reference*, SC26-4399
- *Common Programming Interface Communications Specification*, SC31-6180.

# Index

## A

# B



# C

# D

# E

# F



# G



# H



# I

# K

# L

# M

## N



## O



## P

# Q



# R



# S

# T

# U



# V

# W



# X

# Communicating Your Comments to IBM

AS/400 Advanced Series
APPC Programming
Version 4

Publication No. SC41-5443-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - United States and Canada: 1-800-937-3430
  - Other countries: 1-507-253-5192
- If you prefer to send comments electronically, use this network ID:
  - IBMMAIL(USIB56RZ)
  - IDCLERK@RCHVMW2.VNET.IBM.COM

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

# Readers' Comments — We'd Like to Hear from You

**AS/400 Advanced Series**
**APPC Programming**
**Version 4**

**Publication No. SC41-5443-00**

**Overall, how satisfied are you with the information in this book?**

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|---|---|---|---|---|---|
| Overall satisfaction | □ | □ | □ | □ | □ |

**How satisfied are you that the information in this book is:**

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|---|---|---|---|---|---|
| Accurate | □ | □ | □ | □ | □ |
| Complete | □ | □ | □ | □ | □ |
| Easy to find | □ | □ | □ | □ | □ |
| Easy to understand | □ | □ | □ | □ | □ |
| Well organized | □ | □ | □ | □ | □ |
| Applicable to your tasks | □ | □ | □ | □ | □ |

**Please tell us how we can improve this book:**

Thank you for your responses. May we contact you? □ Yes □ No

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

**Readers' Comments — We'd Like to Hear from You**
SC41-5443-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM CORPORATION
ATTN DEPT 245
3605 HWY 52 N
ROCHESTER MN 55901-7829

Fold and Tape | **Please do not staple** | Fold and Tape



Cut or Fold
Along Line

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC41-5443-00

*Spine information:*

AS/400 Advanced Series APPC Programming *Version 4*