Intrasystem Communications Programming

Version 4
Take Note!

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

August 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 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: 800+937-3430
Other countries: (+1)+507+253-5192

If you have access to 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.

© Copyright International Business Machines Corporation 1997. All rights reserved.
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.
## Contents

| Notices                                      | v       |
| Programming Interface Information           | v       |
| Trademarks and Service Marks                | v       |
| About Intrasystem Communications Programming (SC41-5447) | vii     |
| Who Should Use This Book                    | vii     |
| Prerequisite and Related Information         | vii     |
| Information Available on the World Wide Web | v       |
| Chapter 1. Introduction to Intrasystem Communications | 1-1     |
| Overview of Intrasystem Communications       | 1-1     |
| Using Intrasystem Communications to Test Communications Applications | 1-2     |
| Chapter 2. Configuring Intrasystem Communications | 2-1     |
| Defining the Intrasystem Communications Configuration | 2-1     |
| Example                                      | 2-1     |
| Chapter 3. Running Intrasystem Communications Support | 3-1     |
| Vary On and Vary Off Support                | 3-1     |
| Example                                      | 3-1     |
| Chapter 4. Writing Intrasystem Application Programs | 4-1     |
| Intersystem Communications Function File    | 4-1     |
| Specifying the Program Device Entry Commands | 4-2     |
| Communications Operations                   | 4-3     |
| Starting a Session                          | 4-3     |
| Open/Acquire Operation                      | 4-3     |
| Starting a Transaction                      | 4-3     |
| Evoke Function                               | 4-3     |
| Sending Data                                 | 4-4     |
| Write Operation                              | 4-4     |
| Force-Data Function                          | 4-4     |
| Confirm Function                             | 4-4     |
| Format-Name Function                         | 4-4     |
| Subdevice Selection Function                 | 4-4     |
| End-of-Group Function                        | 4-4     |
| Function-Management-Header Function          | 4-4     |
| Receiving Data                               | 4-4     |
| Read Operation                               | 4-4     |
| Invite Function                              | 4-4     |
| Read-from-Invited-Program-Devices Operation  | 4-5     |
| Waiting for a Display File, an ICF File, and a Data Queue | 4-5     |
| Notifying the Remote Program of Problems     | 4-5     |
| Fail Function                                | 4-5     |
| Cancel Function                              | 4-5     |
| Negative-Response Function                   | 4-5     |
| Using Additional Functions/Operations        | 4-6     |
| Respond-to-Confirm Function                  | 4-6     |
| Request-to-Write Function                    | 4-6     |
| Allow-Write Function                         | 4-6     |
| Cancel-Invite Function                        | 4-6     |
| Timer Function                               | 4-6     |
| Get-Attributes Operation                     | 4-6     |
| Ending Transactions                          | 4-6     |
| Detach Function                              | 4-6     |
| Ending Sessions                              | 4-7     |
| Release Operation                            | 4-7     |
| End-of-Session Function                      | 4-7     |
| Close Operation                              | 4-7     |
| Using Response Indicators                    | 4-7     |
| Receive-Confirm                              | 4-7     |
| Receive-End-of-Group                         | 4-7     |
| Receive-Function-Management-Header           | 4-8     |
| Receive-Fail                                 | 4-8     |
| Receive-Cancel                               | 4-8     |
| Receive-Negative-Response                    | 4-8     |
| Receive-Turnaround                           | 4-8     |
| Receive-Detach                               | 4-8     |
| Using the Input/Output Feedback Area         | 4-8     |
| Using Return Codes                           | 4-8     |
| Chapter 5. Considerations for Intrasystem Communications | 5-1     |
| Application Considerations                   | 5-1     |
| General Considerations                       | 5-1     |
| Open/Acquire Considerations                  | 5-1     |
| Input Considerations                         | 5-1     |
| Confirm Considerations                       | 5-2     |
| Release, End-of-Session, and Close Considerations | 5-2     |
| Performance Considerations                   | 5-2     |
| Prestarting Jobs for Program Start Requests  | 5-2     |
| Appendix A. Language Operations, Data Description, Specifications Keywords, and System-Supplied Formats | A-1     |
| Language Operations                          | A-1     |
| Data Description Specifications Keywords      | A-2     |
| System-Supplied Formats                      | A-2     |
| Appendix B. Return Codes, Messages, and Sense Codes | B-1     |
| Return Codes                                 | B-1     |
| Major Code 00                                | B-1     |
| Major Code 02                                | B-4     |
| Major Code 03                                | B-7     |
| Major Code 04                                | B-8     |
| Major Codes 08 and 11                        | B-9     |
| Major Code 34                                | B-10    |
| Major Code 80                                | B-11    |
| Major Code 81                                | B-13    |
| Major Code 82                                | B-14    |
| Major Code 83                                | B-19    |
| Failed Program Start Requests                | B-24    |
Appendix C. Using Intrasystem Communications to Test Applications ........................................ C-1
Using Intrasystem Communications for Advanced Program-to-Program Communications .......... C-1
Using Intrasystem Communications for Asynchronous Communications ........................................ C-2
Using Intrasystem Communications with Binary Synchronous Communications Equivalence Link ........ C-2
Using Intrasystem Communications for Finance Communications .................................................. C-3
Using Intrasystem Communications for Retail Communications ..................................................... C-4
Using Intrasystem Communications for Systems Network Architecture Upline Facility ................. C-5

Appendix D. Program Examples ........................................ D-1
Description of the Single-Session Inquiry Program Example ......................................................... D-1
ILE C/400 Source Program for a Single-Session Inquiry ................................................................ D-1
ILE C/400 Target Program for a Single-Session Inquiry ................................................................ D-9
Description of the Two-Session Inquiry Program Example .............................................................. D-14
COBOL/400 Source Program for a Two-Session Inquiry .................................................................. D-14
COBOL/400 Target Program for a Two-Session Inquiry .................................................................. D-33
RPG/400 Source Program for a Two-Session Inquiry ..................................................................... D-39
RPG/400 Target Program for a Two-Session Inquiry ..................................................................... D-56

Bibliography ........................................ H-1
AS/400 Books ........................................ H-1
System/36 Communications Books ........................................ H-1

Index ........................................ X-1

Figures

1-1. Overview of Intrasystem Communications ........................................ 1-1
1-2. A Source Program Communicating with Multiple Target Programs ........... 1-2
A-1. Language Operations ........................................ A-1
A-2. Valid Operations for Programming Languages ......................................................... A-1
B-1. Actions for Return Code 0000 ........................................ B-2
B-2. Reason Codes for Rejected Program Start Requests .................................................... B-25
D-1. DDS Source for a Single-Session Source Program Using SRCICFF ................ D-2
D-2. DDS Source for a Single-Session Source Program Using DSPFIL ................ D-2
D-3. Source Program Example — CSRCPGM ........................................ D-4
D-4. DDS Source for a Single-Session Target Program Using TGTICFF ................ D-9
D-5. DDS Source for a Single-Session Source Program Using CUSMSTP .......... D-9
D-6. DDS Source for a Single-Session Target Program Using LGCMSTF ........ D-9
D-7. Target Program Example — CTGTPGM ........................................ D-11
D-8. DDS Source for a Two-Session Source Program Using INTFIL ................. D-15
D-9. DDS for Source Program Two-Session Inquiry Using DSPFIL ................ D-16
D-10. Source Program Example — CSDINT ........................................ D-20
D-11. DDS Source for a Two-Session Target Program Using CF1E .............. D-34
D-12. DDS Source for a Two-Session Target Program Using PF1E .............. D-34
D-13. Target Program Example — CTAIN (User-Defined Formats) ............ D-35
D-14. DDS Source for a Two-Session Source Program Using INTFIL .......... D-39
D-15. DDS Source for Source Program Two-Session Inquiry Using DSPFIL .... D-40
D-16. Source Program Example — RSDINT ........................................ D-44
D-17. DDS Source for an ICF File Used by a Target Program ......................... D-56
D-18. DDS Source for a Database File Used by a Target Program ................ D-57
D-19. Target Program Example — RTDINT ........................................ D-58
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 the 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 USA

This publication could contain technical inaccuracies or typographical errors.

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

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.

Programming Interface Information

This book is intended to help the customer develop communications between two application programs on the same system. It contains information about the intrasystem communications support provided by the AS/400 system. The Intrasystem Communications Programming book contains no programming interfaces for customers.

Trademarks and Service Marks

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

Application System/400
AS/400
COBOL/400
IBM
ILE C/400
ILE COBOL/400
ILE RPG/400
RPG/400
Operating System/400
OS/400
400
Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

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

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

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.
This book contains information about the intrasystem communications support provided by the AS/400 system. It is intended to be used as a guide for developing communications between two application programs on the same system.

For a list of related publications, see the Bibliography.

Who Should Use This Book

This book is intended for the AS/400 application programmer responsible for defining or using intrasystem communications support. It is used for developing application programs that use the support.

You should be familiar with the following information:

- Managing jobs and messages on the AS/400 system, described in the System Operation book, SC41-4203.
- Using the intersystem communications function (ICF) file, described in the ICF Programming book, SC41-5442.
- Communications configuration information described in the Communications Configuration book, SC41-5401.

Prerequisite and Related Information

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

- 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. The AS/400 Information Directory is shipped with the OS/400 operating system at no charge.

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 uniform 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 Intrasystem Communications

AS/400 intrasystem communications allows two application programs, which are running in two different jobs on the same system, to communicate with each other through an ICF file. Using intrasystem communications can help you debug the programs before they are used to communicate with a remote system over a communications line. AS/400 application programs can be written in the ILE C/400*, ILE COBOL/400, or ILE RPG/400* programming languages to use intrasystem communications.

The intrasystem communications support uses intersystem communications function (ICF)\(^1\) data management to handle the sending and receiving of data between the two programs. For communications to begin between programs, the intrasystem communications device description first needs to be configured and varied on.

**Note:** Because intrasystem communications supports process-to-process communications within the same system without the use of communications lines, line and controller descriptions are not used.

**Overview of Intrasystem Communications**

Figure 1-1 provides an overview of the Operating System/400* (OS/400*) intrasystem communications support. Application program A communicates with application program B. ICF data management handles the communications functions and data from your program. The intrasystem communications support handles the communications protocol needed for data transfer and communications between the two programs.

Both the source program (Program A) and the program with which it is communicating (Program B) must use the same device description.

Figure 1-2 on page 1-2 shows how multiple target programs can communicate with the same source program.

When using intrasystem communications, a source program can acquire more than one session for a given device description, and can issue more than one evoke function to start multiple target programs. This means, for example, that PGMA can establish a transaction with PGMB on one session and another transaction with PGMC on another session, and have all the transactions at the same time. However, having established a communications transaction with PGMB on a given session, PGMA cannot then establish a transaction with PGMC on the same session.

**Note:** The term target program is used in this book to refer to the program with which the source program communicates, even though the target program is not on a remote system.

Intrasystem communications imposes no restrictions as to the maximum number of sessions that can be associated with a device. However, the maximum program device (MAXPGMDEV) parameter on the Create ICF File (CRTICFF) command specifies the maximum number of program devices that you can use with the ICF file.

---

\(^1\) The intersystem communications function (ICF) is a function of the operating system that allows a program to communicate interactively with another program or system.
Using Intrasystem Communications to Test Communications Applications

Intrasystem communications can be used to test new communications programs to be run using other communications types. Using intrasystem communications can help you debug the programs before they are used to communicate with a remote system over a communications line.

During testing, you can check only those return codes returned by intrasystem communications and issue only those operations supported by intrasystem communications.

It is also important to note that there are differences in the way intrasystem communications supports certain read and write operations and the way other communications types support such operations. For example, intrasystem communications implicitly responds to a confirm request in the AS/400 system environment, whereas advanced program-to-program communications (APPC) does not. For further details about using intrasystem communications to test other communications types, refer to Appendix C.
Chapter 2. Configuring Intrasystem Communications

This chapter describes the commands used for configuring intrasystem communications on your system.

When using intrasystem communications 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 book.

In this chapter, the parameters of the CL commands that apply to intrasystem communications are described.

### Defining the Intrasystem Communications Configuration

A configuration for intrasystem communications consists of an intrasystem communications device description. The device description describes the characteristics of the logical connection between the two programs. Because programs are communicating with each other on the same system, line and controller descriptions are not supported. To use the intrasystem communications device, it must first be configured and varied on. You can create or change an intrasystem communications device description using the following commands:

- Create Device Description (Intrasystem) (CRTDEVINTR) command
- Change Device Description (Intrasystem) (CHGDEVINTR) command

The parameters for the CRTDEVINTR and CHGDEVINTR commands are:

**DEVD**
Specifies the name for the device description.

**RMTLOCNAME**
Specifies the remote location name with which your program communicates. This parameter cannot be specified on the CHGDEVINTR command.

**ONLINE**
Specifies if this device should be automatically varied on during an initial program load (IPL).
- **YES:** This device is varied on automatically at IPL.
- **NO:** This device is not varied on automatically at IPL. This is the default value.

**AUT**
Specifies the authority you are granting users who do not have specific authority to the object, are not on the authorization list, or whose group has no specific authority to the object.

- **LIBCRTAUT:** The system determines the authority for the object by using the value specified on the CRTAUT parameter on the CRTLIB command for the library containing the object to be created. If the value specified on the CRTAUT parameter is changed, the new value will not affect any existing objects. This is the default value.
- **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 displaying the object. The user is prevented from changing the object. Use authority provides object operational authority and read authority.
- **EXCLUDE:** Exclude authority prevents other users from accessing the object.

**TEXT**
Specifies text that briefly describes the object.

- **BLANK:** No text is specified. This is the default value.
- **description:** Specify no more than 50 characters, enclosed in apostrophes, provided you do not use the prompt screen.

### Example

```
CRTDEVINTR DEVD(INTRALOC) RMTLOCNAME(INTRARMT) ONLINE(*YES) AUT(*CHANGE) TEXT('This is an intrasystem device description')
```

This command creates a device named INTRALOC and a remote location name INTRARMT, allowing two programs to communicate within the same system.
Chapter 3. Running Intrasystem Communications Support

This chapter contains the information you need to run the intrasystem communications support.

Vary On and Vary Off Support

Once an intrasystem communications device has been configured, you can use the Vary Configuration (VRYCFG) command to activate and deactivate the device configuration. This can also be done from the WRKCFGSTS display.

Use the VRYCFG command and specify CFGTYPE(*DEV) and STATUS(*ON) to vary on the configured device description.

Use the VRYCFG command and specify CFGTYPE(*DEV) and STATUS(*OFF) to vary off the configured device description.

The following parameters are applicable to intrasystem communications:

**CFGOBJ**
- Specifies the name of the description for the device to be varied on or off.

**CFGTYPE**
- Specifies the type of configuration description to be varied on or off. This is a required parameter. The only valid entry for intrasystem communications is:
  - **DEV**: Device configuration

**STATUS**
- Specifies the status of the configuration object.
  - **ON**: The object is varied on.
  - **OFF**: The object is varied off.

**RANGE**
- Specifies what configuration elements should be varied, either the configuration element specified (*OBJ) or the configuration element specified and its attached configuration elements (*NET). Devices are considered not to have attached configuration elements. For devices, you can specify either RANGE(*OBJ) or RANGE(*NET).

**VRYWAIT**
- Specifies whether the Ethernet, token-ring, X.25, or switched SDLC, BSC, or asynchronous line description is varied on asynchronously or synchronously. Specify how long the system waits for vary on to be completed (for synchronous vary on) after which the communications file is opened and the session is acquired.

If the VRYWAIT parameter is specified on the VRYCFG command for a line description that is not Ethernet, token-ring, X.25, or switched SDLC, BSC, or asynchronous, the parameter is accepted but ignored.

- **CFGOBJ**: The VRYWAIT parameter value specified in the line description is used.
- **NOWAIT**: The system does not wait for vary on completion. The line is varied on asynchronously.

**ASCVRYOFF**
- Specifies whether the vary off is asynchronous. This parameter is not allowed when STATUS(*ON) is specified.
  - **NO**: The vary off is synchronous.
  - **YES**: The vary off is asynchronous.

**Example**

VRYCFG CFGOBJ(INTRALOC) CFGTYPE(*DEV) STATUS(*ON) RANGE(*OBJ)

This command varies on the configured device description INTRALOC.
Chapter 4. Writing Intrasystem Application Programs

This chapter describes how an application program uses the intersystem communications function (ICF) file and the intrasystem communications support. The program can be coded using the ILE C/400, C Set ++, ILE COBOL/400, or ILE RPG/400 programming languages, which allows the program to do the following functions:

- Start a session by opening an ICF file and acquiring a program device.
- Send and receive information by writing or reading to an ICF file.
- End a session by releasing the program device and closing the ICF file.

This chapter also includes a description of the read and write operations that specify a record format containing specific communications functions. Record formats can be defined using data description specifications (DDS), or you may use system-supplied formats.

After an operation is completed, a return code (and a high-level language file status) is returned to your application. The return code indicates whether the operation was completed successfully or unsuccessfully. Along with the return code, exception messages may also be issued. Refer to Appendix B for more information about return codes and to the appropriate language reference books for more information about the high-level language file status.

Intersystem Communications Function File

An intersystem communications function (ICF) file must be created before your application can use the intrasystem communications support. The ICF file is used to describe how data is presented to the program with which your program is communicating, and how data is received from that program. If you are using DDS keywords, use the Create Intersystem Communications Function File (CRTICFF) command to create an ICF file. If you are using the system-supplied formats (such as $$SEND), you do not need to create an ICF file. The ICF file QICDMF, which is in the library QSYS, is supplied by IBM for communications.

The ICF file is a system object of type *FILE with a specific user interface. This interface is made up of a 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 used to manage the ICF file, and are described in detail in the ICF Programming book.

- CRTICFF: Create ICF File. This command allows you to create an ICF file and file level attributes.
- 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 to an ICF file.
- 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 do the following:
  - Temporarily add the program device entry, the remote location information, and the session-level attributes to the ICF file.
  - Temporarily change a program device entry with the specified remote location information and session-level attributes for an ICF file. These changes are only in effect for the job.
- RMVICFDEVE: Remove ICF Device Entry. This command allows you to permanently remove the program device entry previously added to an ICF file 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 valid values for each parameter for intra-system communications.

FILE

Specifies the name and library of the ICF file to which you are adding or changing the program device entry. The FILE parameter is not available on the OVRICFDEVE command.

*LIBL: Intrasystem communications support uses the library list to locate the ICF file.

*CURLIB: Intrasystem communications support uses the current library for the job to locate the ICF file. If no current library entry exists in the library list, intrasystem communications uses QGPL.

filename: A 1- to 10-character value that specifies the name of the ICF file.

library-name: A 1- to 10-character value that specifies the library where the ICF file is located.

PGMDEV

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

pgm-device-name: Enter a 1- to 10-character value for the program device name being defined. This name is used on device-specific input and output operations to identify the program device and the attributes.

RMTLOCNAME

Specifies the remote location name with which your program communicates. A remote location name must be specified on the ADDICFDEVE command or an OVRICFDEVE command. If a remote location name is not specified, a major and minor error code are returned when the program device is acquired.

*REQUESTER: The name used to refer to the communications device through which the program was started. The session that is assigned when the program device is acquired is the same session that receives the program start request. If the program is not started as a result of a program start request, the acquire operation for the program device fails. The target program always uses *REQUESTER as the remote location name in the ICF file to connect to the session that the source program uses to send the program start request.

remote-location-name: Enter a 1- to 8-character name for the remote location name that should be associated with the program device.

FMTSLT

Specifies the type of record format selection used for input operations for all devices.

*PGM: The program determines what record formats are selected. If an input (read) operation with a record format name is specified, that format is always selected. If an input operation without a record format is specified, the default format (the first record format in the file) is always selected. This also means that if any record identification (RECID) keywords are specified in the data description specifications (DDS) for the file, they are not taken into consideration when the record is selected.

*RECID: The RECID keywords specified in DDS for the file are used to specify record selection. If no RECID keywords are specified in the file, an error message is sent and an acquire operation for the program device will fail.

*RMTFMT: The remote format names received from the sending program are used to select the record format.

CMNTYPE

Identifies the communications type for which you define a program device entry. You should specify the value *INTRA or *ALL for this parameter.

*INTRA: The prompt for all intrasystem communications-supported attributes.

*BATCH

Specifies if batch processing is performed for the session. If you specify RMTLOCNAME(*REQUESTER), this parameter is ignored. The program that issues the evoke function determines whether batch processing can occur.

*NO: Specifies that batch processing does not occur.

*YES: Specifies that batch processing occurs.

SECURE

The SECURE parameter is valid only on the OVRICFDEVE command. This parameter does not apply to the ADDICFDEVE or CHGICFDEVE commands. This parameter is used to restrict the effects of override processing.

*NO: Specifies no protection from other program device overrides.

*YES: Specifies program device override protection from override commands started in earlier programs.
Communications Operations

This section provides a description of the operations you can code into a program that uses intrasystem communications support to communicate with another program.

Starting a Session

A communications session is a logical connection by which a program running in one job can communicate with another program running in a different job. A communications session is established with an acquire operation, and is ended with a release operation or end-of-session function.

Open/Acquire Operation

Your application program uses the acquire operation to establish the session on which your program will communicate with another program. Intrasystem communications uses the value of each parameter that was specified on the ADDICFDEVE or OVRICFDEVE command for the program device.

The parameters are used to determine the following session characteristics:

- **Format selection option**: This indicates the type of processing that needs to be done to determine what record format to use on an input operation. Intrasystem communications supports all three format selection options: *PGM, *RECID, and *RMTFMT.

- **Batch option**: This indicates whether or not batch processing is performed for the session. The following functions are only valid if BATCH(*YES) is specified on the ADDICFDEVE or OVRICFDEVE command:
  - Function-management-header function
  - Cancel function
  - Negative-response function

Starting a Transaction

A transaction is a logical connection between two programs. Use an evoke function to start a transaction between your program and another program.

Evoke Function

Your program uses the evoke function to start a transaction with a target program after you start a session. It is not valid if your program is already communicating with another program on the same session.

The program that issues an evoke function (the source program) is initially in send state unless, for example, it also issues a read operation, an allow-write function, or an invite function to the other program. The source program would then be in receive state and the other program can send data.

The program with which the source program communicates (the target program) is initially in receive state, and should issue read operations until a receive-turnaround indication is received.

The program that is in send state controls the transaction, and determines what the other program must do. For example, if a program that is in send state sends data, then the program in receive state should issue a read operation to receive the data (it cannot also send data).

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

- The name of the program with which your program is to communicate
- The library in which the other program exists (optional)
- User-defined program initialization parameters (optional)
- Synchronization level (optional)
- Security information (optional)

If your program is using the EVOKE DDS keyword, 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 for synchronization level. In this case, synchronization or confirmation is not allowed.

If you specify program initialization parameters (PIP) with the evoke function, each parameter that is sent should be equal in length to the corresponding parameter specified in the target program. If it is longer than the parameter length in the target program, truncation occurs. If it is shorter than the parameter length in the target program, results that are not predictable may occur.

For information on how to code the evoke function, refer to the ICF Programming book and the DDS Reference book.

---

1 The program initialization parameter (PIP) is the initial parameter value passed to a target program as input or used to set up the process environment.
Sending Data

You can send data during a transaction using the write operation. With the write operation, you can, for example, specify the end of a group of records, or indicate when your program has finished sending data.

Write Operation

The write operation is used to send data records to another program. Each write operation sends only one data record. Intrasystem communications supports various functions that are discussed following this description. These functions may be issued by your program to another program either with or without data. The only exception is the function-management-header function, which requires data to be sent.

Only one write operation can be issued at a time. Therefore, if your program issues a second write operation while the first one has not been received by the other program, the second write operation will not complete until the data previously sent is read by the other program.

Force-Data Function

Your program uses the force-data function to send communications data currently held in the output buffer. However, because intrasystem communications does not buffer data, the force-data function does not provide any additional function.

Confirm Function

Your program uses the confirm function to indicate to the other program that a response is needed before an operation can complete. Once a confirm function is issued, control does not return to your program until the other program with which you are communicating responds to the confirm.

Note: The confirm function is allowed only if the transaction was started with a synchronization level of confirm on the evoke function.

Format-Name Function

Your program uses the format-name function to send the record format name, along with data, to the other program. This is the record format name that should be used by the other program on the corresponding read operation.

Note: The value *RMTFMT for the FMTSLT parameter needs to be specified by the other program on the ADDICFDEV or OVRICFDEV command for the record format name to be used.

Subdevice Selection Function

Your program uses the subdevice selection function to specify to the other program the device to which the output is to be directed. Intrasystem communications sends the subdevice code as a separate record before sending the data.

Note: The subdevice function is ignored on any write operation except for the first operation in a group.

End-of-Group Function

Your program uses the end-of-group function to indicate to the other program that this is the last record in a group. It does not indicate, however, that your program is ready to receive data. The session remains in a send state.

Function-Management-Header Function

Your program uses the function-management-header function to indicate to the other program that function-management-header data is being sent. Function-management-header data contains control information for the data that is to follow, and is only valid under the following conditions:

- If the program is running in batch mode (BATCH(*YES) was specified on the ADDICFDEV or OVRICFDEV command)
- If data is being sent (the data length is greater than zero)

Intrasystem communications inserts the characters FMH before the data being sent.

Receiving Data

Intrasystem communications supports various functions designed to obtain data from the other program.

Read Operation

Your application program uses the read operation to obtain data or control information from the other program. This operation causes the user to wait for the data if it is not immediately available. You can use the read operation by itself or in combination with other write operations, in which case the write operation is performed first, followed by the read operation.

Invite Function

Your application program uses the invite function to request input data from another program, but it receives control without waiting for the input. To obtain the data, you must issue either a read or a read-from-invited-program-devices operation. This function can be issued by itself or in combination with other write operations.
Read-from-Invited-Program-Devices Operation

You can use the read-from-invited-program-devices operation to obtain data from any program that has responded to an invite function 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.

Waiting for a Display File, an ICF File, and a Data Queue

Use data queues when a program must wait for a display file, an ICF file, and a data queue, in any combination, at the same time. The following commands are used with the specified 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)

Use these commands to indicate a data queue that will have entries placed in it when one of the following occurs:

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

For more information, see the CL Programming book and the ICF Programming book.

Notifying the Remote Program of Problems

Your program uses the fail, cancel, and negative-response functions to indicate that an error has occurred during a transaction with the target program.

Fail Function

Your program issues the fail function to indicate that it has detected an error in the data while it was sending or receiving data. The fail function can be sent in either send or receive state. No data can be sent with the fail function.

If a program that is in the send state issues a fail function, the last record before the fail function was issued is still sent to the other program.

If a program is in the receive state and issues a fail function, intrasystem communications support discards the incoming data, informs the other program that a fail is being sent by returning a 0402 return code, and changes the state of your program's session from receive to send state.

In either case, the program that issued a fail function should send, and the program that received the fail must receive.

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 receives a fail return code.

Cancel Function

Your application program can issue a cancel function to indicate that it detected an error in the data it was sending. The cancel function is only valid under the following conditions:

- If your program is running in batch mode (BATCH(*YES) was specified on the ADDICFDEVE or OVRICFDEVE command)
- Within a group of records
- When the program is in send state

When your program is sending data and issues a cancel function, intrasystem communications support informs the other program that a cancel is being sent. No data may be sent with a cancel operation.

The program that issues a cancel function should send, and the program that receives the cancel must receive.

Issuing a cancel function is similar to issuing a fail function when your program is sending data.

Negative-Response Function

Your application program can use a negative-response function to indicate that it detected an error in the data it was receiving. The negative-response function is only valid under the following conditions:

- If your program is running in batch mode (BATCH(*YES) was specified on the ADDICFDEVE or OVRICFDEVE command)
- Within a group of records, or as the first function after receiving an end-of-group function
- When the program is in receive state, but an invite function has not yet been issued or is currently not in effect

When your program issues a negative-response function, intrasystem communications support discards any data being received, and informs the other program that a negative-response is being sent.
Your program can also send eight bytes of **sense data** with the negative-response function to inform the other program about the reason for the error. Intrasystem communications checks this data to ensure that the first four bytes are 10xx, 08xx, or 0000, where x is a digit. If not, the function is rejected, with a return code of 831B. If your program does not supply sense data, then intrasystem communications sends the code 08110000.

Issuing a negative-response function is similar to issuing a fail function when your program is receiving data.

### Using Additional Functions/Operations

Intrasystem communications supports the following additional functions or operations.

#### Respond-to-Confirm Function

Your program uses the respond-to-confirm function to send a positive response to a received confirm request. The positive response indicates that data was received without error or that the request received may be performed (such as a detach).

You can issue the respond-to-confirm function only after receiving a confirm request from the other program.

A respond-to-confirm function is not required, however, and an implicit positive response is sent if the next operation is not a fail, cancel, negative-response, or end-of-session function.

#### Request-to-Write Function

Your program uses the request-to-write function to indicate that it wants to send something to the other program rather than continue receiving data. The other program decides, however, whether to stop sending data and when it will stop.

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

Your program can issue the request-to-write function only when no invite function is in effect, and only when your program is in the receive state.

#### Allow-Write Function

Your program issues the allow-write function to inform the program with which it is communicating that it is finished sending data and is ready to receive.

Intrasystem communications sends data and an indication to the other program that allow-write is being sent. If the operation is successful, a return code of 0001 is returned to indicate that your program is ready to receive data.

#### Cancel-Invite Function

Your program uses the cancel-invite function to attempt to cancel an outstanding invite function for which no data has been received. Cancel-invite is only valid when an invite function is still in effect.

When your program issues a cancel-invite function, intrasystem communications determines if data has been received from the other program. If no data has been received, the invite is canceled, and your program is changed from an invite state to send state.

If data has already been received from the other program, the invite is not canceled, and a return code of 0412 is returned. Your program must then issue a read or read-from-invited-program-devices operation to receive the data that the other program has already sent.

#### Timer Function

Your program can use the timer function to set the maximum amount of time your program waits to receive data when issuing the read-from-invited-program-devices operation.

#### 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.

### Ending Transactions

The detach function is used to end an active transaction between your program and the program with which it is communicating.

#### Detach Function

Your program uses the detach function to inform the other program that your program is finished sending data and wants to end the transaction.

Intrasystem communications sends the data and indicates to the other program that the current record is the last record.

When a detach function is issued with a confirm function, the transaction is ended by your program if a positive response is received, and no further input or output operations with the other program is allowed. When a detach function is issued without a confirm function, the transaction ends without waiting for a response from the other program. When the target program receives the detach, it can no longer communicate with the source program and must end the logical con-
connection to the session by ending the session. A source program must issue an evoke function to establish communications again with a target program after sending or receiving a detach function.

When a detach function is issued by a target program, its logical connection to the session, as well as to the transaction, is ended.

Ending Sessions

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

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 session is associated with the source program, the release operation ends the session immediately (unless some error condition occurs). The operation frees the resources (allocated to the program) 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 only valid when a transaction is not active.

- If the session is associated with the target program, the release operation only temporarily ends the connection to the source program. The session is kept active, and is not available for use by other programs until the target program issues an end-of-session function or ends. If a detach has not been done (that is, the transaction is still active), an acquire can be issued to continue communications on that session.

End-of-Session Function

Your program uses the end-of-session function to end a session with another program. Unlike the release operation, the end-of-session function always ends the session. However, if the function is issued during an active transaction, intrasystem communications abnormally ends the session.

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

Close Operation

Your program uses the close operation to close the ICF file and to end the program’s attachment to any active session the program has acquired. If the close operation is issued to a session that was established by a source program, intrasystem communications ends the session and deallocates all resources that were allocated for the file. 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 to a session associated with a target program, the connection to the program is only temporarily ended. The session is kept active and is not available for use by other programs until the target program issues an end-of-session function or ends.

Using Response Indicators

Response indicators\(^2\) 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 response indicator to receive an indication from the other program that the record it received contained a confirm request. A received confirm request indicates the other program is expecting your program to perform a specific action to synchronize the programs. This action can be a respond-to-confirm function to respond positively or a fail or end-of-session function to respond negatively. Your program can also do a normal input/output operation to respond positively.

The presence of the confirm request is also indicated by the minor return codes 14, 15, 17, 1C, 44, 45, and 47 with the major return code 00 (user data received) or 02 (user data received but program is being ended), or by the minor return codes 14, 15, 17, and 1C with the major return code 03 (no data received).

Receive-End-of-Group

The receive-end-of-group response indicator is used to indicate that the other program has sent the last record in a group.

The presence of the end-of-group function is also indicated by the minor return codes 03, 07, 17, and 47 with the major return code 00 (user data received) or 02 (user data received but program is being ended), or by the minor return codes 03, 07, 17, and 1C with the major return code 03 (no data received).

---

\(^2\) A response indicator is a 1-character field passed with an input record from the system to a program to provide information about the data record.
but program is being ended), or by the minor return codes 03 and 17 with the major return code 03 (no data received).

**Receive-Function-Management-Header**

Your program uses the receive-function-management-header response indicator to receive an indication from the other program that function-management-header data was received. The first three characters of the received data are the characters FMH.

The presence of function-management-header data is also indicated by the minor return codes 04, 05, 07, 44, 45, and 47 with the major return code 00 (user data received) or 02 (user data received but program is being ended).

**Receive-Fail**

Your program uses the receive-fail response indicator to receive an indication that the other program encountered an error when it was sending or receiving data, and your program should take the appropriate recovery action. Your program remains in receive state after receiving the receive-fail indicator and should continue to issue read operations.

Receipt of a fail request is also indicated by the minor return code 02 with the major return code 03 (no data received) or 04 (output exception occurred).

The failure notification is always received without user data.

**Receive-Cancel**

Your program uses the receive-cancel response indicator to receive an indication that the other program encountered an error when it was sending data.

Receipt of a cancel request is also indicated by the minor return codes 30 and 31 with the major return code 83.

The cancel notification is always received without user data.

**Receive-Negative-Response**

Your program uses the receive-negative-response response indicator to receive an indication that the other program encountered an error when it was receiving data.

Your program must issue an input operation to receive the eight character sense code that the other program (or intra-system) sends with the negative-response indication.

Receipt of a negative-response function is also indicated by the 8319 return code. Refer to Appendix B for a description of the return code.

**Receive-Turnaround**

Your program uses the receive-turnaround response indicator to receive an indication from the other program indicating that it is ready to receive data.

The presence of the turnaround indication is also indicated by the minor return codes 00, 04, 14, and 44 with the major return code 00 (user data received) or 02 (user data received but program is being ended), or by the minor return codes 00 and 14 with the major return code 03 (no data received).

**Receive-Detach**

Your program uses the receive-detach response indicator to receive an indication when the received data ends a transaction (the detach request has been received).

The presence of the detach request is also indicated by the minor codes 08 (detach only) and 1C (detach and confirm request) with the major return codes 00 (user data received), 02 (user data received but program is being ended), or 03 (no data received).

**Using the Input/Output Feedback Area**

Your program may have access to the file-dependent input/output (I/O) feedback area. If it does, you should be aware of certain fields when writing applications using intra-system communications:

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

- **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.

- **Request-to-write indicator**
  - This field indicates whether the other program has requested permission to send data.

- **Format name**
  - This field contains the record format name used to receive the data on an input operation.

**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 completed
- The operation that should be issued next
Example: On an input operation, a major return code of 00 indicates that data was received. Along with this major code, intrasystem communications could return one of these minor codes:

- 01: Indicates that your program should continue receiving data.
- 08: Indicates the other 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 the target program, end the session or go to end of job.
- 1C: Indicates the program with which your program is communicating wants to end 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 other 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.

Example, Error Condition: Another example would be a major code of 83. In this case, an error was detected that may be recoverable. Different minor codes can be returned, just as for the 00 major. For example, if your program receives a CD minor return code, your program has issued a confirm function that is currently not allowed. Your program is using a transaction that was not started with a synchronization level of confirm. 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.

It is recommended that your program check the ICF return codes at the completion of every operation to ensure that the operation completed successfully or that the appropriate recovery action can be taken.

Refer to Appendix B for a description of the return codes that can be returned to your application when it is using intrasystem communications.
Chapter 5. Considerations for Intrasystem Communications

This chapter describes the application and performance considerations for intrasystem communications.

Application Considerations

Before writing programs that use intrasystem communications, you must understand some of the characteristics of the AS/400 environment.

General Considerations

These general considerations apply to your program and the program with which it is communicating.

- The first operation following the acquire operation by a source program should be a write operation with an evoke function specified. The evoke function starts the program with which the source program is going to communicate.
- The source program can send program initialization parameters, with the evoke function, to the other program only if the other program supports the receipt of these parameters.
- 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 (requesting program device).
- When a program is in 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 can also be used if your program is to send an error condition to the other program.
- When a program is in send state, all operations except open (to the opened ICF file), acquire (to the same program device), negative-response, and the request-to-write function are generally valid.
- When a program is receiving data, it should continue to issue input operations until one of the following is received:
  - A minor return code indicating that your program may now send data. The RCVTRNRND keyword can also be used.
  - A minor return code indicating that detach has been received. The RCVDETACH keyword can also be used.
  - A major and minor return code indicating an error condition, for example, any of the 80 major return codes.
  - Data that contains an error, in which case the program should issue a negative-response or fail function.

To increase performance, two communicating programs can change states implicitly without using the turnaround indication by synchronizing their input and output operations. For example, if Program A is sending data and issues a read operation to Program B, which is receiving data, Program B can issue a write operation without having received a turnaround indication. Program A is then ready to receive data and Program B can send data.

Open/Acquire Considerations

The following information describes how the acquire operation is used to start a session between the source program (Program A) and the program with which it is communicating (Program B).

- If Program B acquires a program device, other than the requesting program device, a new session is established and the connection with the source program (Program A) is not established. No error is indicated because it is valid for Program B to be a target on one session and to be a source program on another session. If the program issues an input operation as the first operation to the newly established session, and an evoke function has not yet been issued, it will receive a return code indicating that no transaction is active.
- Multiple sessions (that run at the same time) can be established with multiple programs. The program device names are used to distinguish the sessions within your program.

Input Considerations

The following information describes the input considerations for your program.

- The receive indicators RCVTRNRND, RCVDETACH, RCVENDGRP, 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 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.
- The actual received data length can be determined from the file-dependent I/O feedback area.
- When a write operation is issued following an invite, the system performs an implicit cancel-invite function and your program can begin sending data to the other program, provided data was not waiting to be received.
- When a read operation is issued following a write, or following a read in which turnaround was received, the system performs an implicit allow-write function and begins waiting for data from the other program.
Confirm Considerations

The following information describes how the confirm function is used by both your program and the other program with which it is communicating.

- Your program requests that the other 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 other program in the following ways:
  - A major return code of 00 or 02, with the minor return codes 14, 15, 17, 1C, 44, 45, or 47, or a major return code of 03 with the minor return codes 14, 15, 17, or 1C.
  - 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 respond-to-confirm function, or issue any input/output operation except the fail, negative-response, or end-of-session function.
- To respond negatively to the request, do the following:
  - Issue a fail function. In this case, your program is responsible for the appropriate level of error recovery.
  - Abnormally end the transaction and session by issuing either an end-of-session function or a close operation.

- When it is essential to your application program that the other program be started before you issue output operations to it, specify the confirm function with the evoke function. The evoke function will not complete until the other program responds to the confirmation request.
- Because the output operation with the confirm function specified waits for a positive or negative response before control is returned to the program, the source and target programs should be coded to minimize the amount of time between receiving the confirm request and sending the response. If the program receiving the confirm request performs complex processing before sending a response, the delay time can be significant.

Performance Considerations

If your program issues more than one evoke function, poor system performance may result. This is because each evoke function that results in a successful transaction causes a job to be started on the system. Because job initializations require a fair amount of system resources, you should design your application to minimize evoke functions.

In general, your program should issue multiple evoke functions only when jobs are long-running or when multiple target jobs need to run at the same time. If you issue a large number of evoke functions in your program, you can use prestart jobs, described on 5-2, to minimize the time required to start a job.

Prestarting Jobs for Program Start Requests

A program start request is a request made by your program to start another program. When your program issues an evoke function, this signals a program start request to the intrasystem communications support.

To minimize the time required to carry out a program start request, you can use the prestart jobs entry to start a job for the other program before it receives the program start request. To use prestart jobs, you need to define both communications and prestart job entries in the same subsystem description, and make certain programming changes to the prestart job program with which your program communicates. For information about how to use prestart jobs, refer to the ICF Programming book.
Appendix A. Language Operations, Data Description, Specifications Keywords, and System-Supplied Formats

This appendix contains charts that show the following for intrasystem communications:

- All valid language operations supported by ICF
- Valid operations for each programming language that supports ICF
- Data description specifications (DDS) processing keywords
- System-supplied formats

## Language Operations

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

### Figure A-1. Language Operations

<table>
<thead>
<tr>
<th>ICF Operation</th>
<th>ILE RPG/400 Operation Code</th>
<th>ILE COBOL/400 Procedure Statement</th>
<th>ILE C/400 Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>OPEN</td>
<td>OPEN</td>
<td>fopen, _Ropen</td>
</tr>
<tr>
<td>Acquire</td>
<td>ACQ</td>
<td>ACQUIRE</td>
<td>_Racquire</td>
</tr>
<tr>
<td>Get attributes</td>
<td>POST</td>
<td>ACCEPT</td>
<td>_Rdevatr</td>
</tr>
<tr>
<td>Read</td>
<td>READ</td>
<td>READ</td>
<td>fread, _Rreadn</td>
</tr>
<tr>
<td>Read-from-invited-</td>
<td>READ1</td>
<td>READ1</td>
<td>_Rreadindv</td>
</tr>
<tr>
<td>program-devices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td>WRITE</td>
<td>WRITE</td>
<td>fwrite, _Rwrite</td>
</tr>
<tr>
<td>Write/Read</td>
<td>EXFMT</td>
<td>Not supported</td>
<td>_Rwriterd</td>
</tr>
<tr>
<td>Release</td>
<td>REL</td>
<td>DROP</td>
<td>_Release</td>
</tr>
<tr>
<td>Close</td>
<td>CLOSE</td>
<td>CLOSE</td>
<td>fclose, _Rclose</td>
</tr>
</tbody>
</table>

A read operation can be directed either to a specific program device or to any invited program device. The support provided by the compiler you are using determines whether to issue an ICF read or read-from-invited-program-devices operation, based on the format of the read operation. For example, if a read is issued with a specific format or terminal specified, the read operation is interpreted as an ICF read operation. Refer to the appropriate language reference book for more information.

Figure A-2 shows all the valid operations for each programming language that supports ICF (ILE C/400, C Set ++ for OS/400, and ILE RPG/400 programming languages).
### Data Description Specifications Keywords

The following lists the data description specifications (DDS) processing keywords that are valid for intrasystem communications.

<table>
<thead>
<tr>
<th>DDS Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALWWRT</td>
<td>Allow-write</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Cancel</td>
</tr>
<tr>
<td>CNLINVITE</td>
<td>Cancel-invite</td>
</tr>
<tr>
<td>CONFIRM</td>
<td>Confirm</td>
</tr>
<tr>
<td>DETACH</td>
<td>Detach (End of transaction)</td>
</tr>
<tr>
<td>ENDGRP</td>
<td>End-of-group</td>
</tr>
<tr>
<td>EOS</td>
<td>End-of-session</td>
</tr>
<tr>
<td>EVOKE</td>
<td>Evoke</td>
</tr>
<tr>
<td>FAIL</td>
<td>Fail</td>
</tr>
<tr>
<td>FMH</td>
<td>Function-management-header</td>
</tr>
<tr>
<td>FMTNAME</td>
<td>Format-name</td>
</tr>
<tr>
<td>FRCDTA</td>
<td>Force-data</td>
</tr>
<tr>
<td>INVITE</td>
<td>Invite</td>
</tr>
<tr>
<td>NEGRSP</td>
<td>Negative-response</td>
</tr>
<tr>
<td>RCVCANCEL</td>
<td>Receive-cancel</td>
</tr>
<tr>
<td>RCVCONFIRM</td>
<td>Receive-confirm</td>
</tr>
<tr>
<td>RCVDETACH</td>
<td>Receive-detach</td>
</tr>
<tr>
<td>RCVENDGRP</td>
<td>Receive-end-of-group</td>
</tr>
<tr>
<td>RCVFAIL</td>
<td>Receive-fail</td>
</tr>
<tr>
<td>RCVFMH</td>
<td>Receive-function-management-header</td>
</tr>
<tr>
<td>RCVNEGRSP</td>
<td>Receive-negative-response</td>
</tr>
<tr>
<td>RCVTRNRND</td>
<td>Receive-turnaround</td>
</tr>
<tr>
<td>RECID</td>
<td>Record-identification</td>
</tr>
<tr>
<td>RQSWRT</td>
<td>Request-to-write</td>
</tr>
<tr>
<td>RSPCONFIRM</td>
<td>Respond-to-confirm</td>
</tr>
<tr>
<td>SECURITY</td>
<td>Security</td>
</tr>
<tr>
<td>SUBDEV</td>
<td>Subdevice</td>
</tr>
<tr>
<td>SYN_LVL</td>
<td>Synchronization level</td>
</tr>
<tr>
<td>TIMER</td>
<td>Timer</td>
</tr>
<tr>
<td>VARLEN</td>
<td>Variable-length data</td>
</tr>
</tbody>
</table>

### System-Supplied Formats

The following lists all the keyword functions performed by the system-supplied formats that are valid for intrasystem communications.

<table>
<thead>
<tr>
<th>System-Supplied *Formats</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$CANL</td>
<td>Cancel with invite</td>
</tr>
<tr>
<td>$$CANLNI</td>
<td>Cancel</td>
</tr>
<tr>
<td>$$CNLINV</td>
<td>Cancel-invite</td>
</tr>
<tr>
<td>$$EOS</td>
<td>End-of-session</td>
</tr>
<tr>
<td>$$EVOK</td>
<td>Evoke with invite</td>
</tr>
<tr>
<td>$$EVOKET</td>
<td>Evoke with detach</td>
</tr>
<tr>
<td>$$EVOKNI</td>
<td>Evoke</td>
</tr>
<tr>
<td>$$FAIL</td>
<td>Fail</td>
</tr>
<tr>
<td>$$NRSP</td>
<td>Negative-response with invite</td>
</tr>
<tr>
<td>$$NRSPNI</td>
<td>Negative-response</td>
</tr>
<tr>
<td>$$RCD</td>
<td>Request-to-write with invite</td>
</tr>
<tr>
<td>$$SEND</td>
<td>Send with invite</td>
</tr>
<tr>
<td>$$SENDE</td>
<td>Send with end-of-group</td>
</tr>
<tr>
<td>$$SENDET</td>
<td>Send with detach</td>
</tr>
<tr>
<td>$$SENDFM</td>
<td>Send function-management-header with invite</td>
</tr>
<tr>
<td>$$SENDNF</td>
<td>Send with function-management-header</td>
</tr>
<tr>
<td>$$SENDNI</td>
<td>Send</td>
</tr>
<tr>
<td>$$TIMER</td>
<td>Timer</td>
</tr>
</tbody>
</table>
Appendix B. Return Codes, Messages, and Sense Codes

Return Codes

This section describes all the return codes that are valid for intrasystem communications. 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 book 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 AS/400 application program that issues the operation and receives a return code from ICF communications. The other program refers to the application program 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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td><strong>Description</strong>: For input operations issued by your program, 0000 indicates that your program received some data with a turnaround indication. The other 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. <strong>Action</strong>: If your program received a turnaround on an input operation, issue an input or output operation. For the actions which can be taken after 0000 is received, refer to the following table:</td>
</tr>
</tbody>
</table>
Figure B-1. Actions for Return Code 0000

<table>
<thead>
<tr>
<th>Type of Session</th>
<th>Last Operation Issued</th>
<th>Actions Your Program Can Take</th>
</tr>
</thead>
<tbody>
<tr>
<td>Started by a source program</td>
<td>Acquire or open</td>
<td>Issue an evoke or timer function, or a get-attributes operation.</td>
</tr>
<tr>
<td></td>
<td>Evoke with detach or write with detach</td>
<td>Issue another evoke function, issue a release operation, continue local processing, or end.</td>
</tr>
<tr>
<td></td>
<td>Any other output operation</td>
<td>Issue another output operation (except evoke), or issue an input operation.</td>
</tr>
<tr>
<td></td>
<td>End-of-Session</td>
<td>Continue local processing or end.</td>
</tr>
<tr>
<td>Started by a remote program start</td>
<td>Acquire or open</td>
<td>Issue an input or output operation.</td>
</tr>
<tr>
<td>request</td>
<td>Write with detach</td>
<td>Continue local processing or end. This session has ended.</td>
</tr>
<tr>
<td></td>
<td>Any other output operation</td>
<td>Issue another output operation (except evoke), or issue an input operation.</td>
</tr>
<tr>
<td>End-Of-Session</td>
<td></td>
<td>Continue local processing or end.</td>
</tr>
</tbody>
</table>

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.

0003 Description: On a successful input operation, your program received some data with an end-of-group indication.

Action: Issue an input operation to receive the next group of records.

0004 Description: On a successful input operation, your program received some data with a function-management-header (FMH) and a turnaround indication. The other program is ready to receive data.

Action: Issue an output operation.

0005 Description: On a successful input operation, your program received some data with a function-management-header (FMH).

Action: Your program can issue another input operation to continue receiving data until it receives a turnaround indication or a detach indication.

0007 Description: On a successful input operation, your program received a function-management-header (FMH) and an end-of-group indication. Your program should continue to receive data.

Action: Issue another input operation to receive the next group of records.

0008 Description: On a successful input operation, your program received a detach indication with the last of the data. The communications transaction with the other program has ended.

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 other program started the transaction, your program can either issue an end-of-session function or end.

0010 Description: On a successful output operation, your program received a request-to-write indication. The other program wants to send data as soon as possible. You should allow the other program to send this data.

Action: Issue an input operation as soon as possible.

0014 Description: On a successful input operation, your program received some data with a turnaround indication. In addition, the other 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 or 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 other 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.

0017 Description: On a successful input operation, your program received some data with an end-of-group indication. In addition, the other 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 another 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 other 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 other 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 other 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.

0044 Description: On a successful input operation, your program received some data with a function-management-header (FMH) and a turnaround indication. In addition, the other 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 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 data with a function-management-header (FMH). In addition, the other 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.

0047 Description: On a successful input operation, your program received some data with a function-management-header (FMH) and an end-of-group indication. In addition, the other 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.
**Major Code 02**

- **Description:** The input operation issued by your program completed successfully. Your program may have received some data or a message from the 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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 0200 | **Description:** On a successful input operation, your program received some data with a turnaround indication. Also, your job is being ended (controlled). The other program is ready to receive data from your program.  

**Action:** Your program can issue an input or 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.  

**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. |
| 0203 | **Description:** On a successful input operation, your program received some data with an end-of-group indication. Also, your job is being ended (controlled).  

**Action:** Your program can issue an input operation to receive the next group of records. 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 data with a function-management-header (FMH) and a turnaround indication. Also, your job is being ended (controlled). The other program is ready to receive data.  

**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 data with a function-management-header (FMH). Also, your job is being ended (controlled).  

**Action:** Your program can issue another input operation to continue receiving data until it receives a turnaround indication or a detach indication. 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. |
0207 Description: On a successful input operation, your program received a function-management-header (FMH) and an end-of-group indication. Also, your job is being ended (controlled).

Action: Your program can issue another input operation to receive the next group of records. 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 other program has ended. 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 other 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.

0214 Description: On a successful input operation, your program received some data with a turnaround indication. In addition, the other 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 or output operation. If your program does detect an error, issue a fail function, or end your program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0215 Description: On a successful input operation, your program received some data. In addition, the other 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.

0217 Description: On a successful input operation, your program received some data with an end-of-group indication. In addition, the other 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 another 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 other 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 other 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 other 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.

**0244 Description:** On a successful input operation, your program received some data with a function-management-header (FMH) and a turnaround indication. In addition, the other program requested confirmation. Also, your job is being ended (controlled).

**Action:** 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 data with a function-management-header (FMH). In addition, the other program requested confirmation. Also, your job is being ended (controlled).

**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. 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.

**0247 Description:** On a successful input operation, your program received some data with a function-management-header (FMH) and an end-of-group indication. In addition, the other program requested confirmation. Also, your job is being ended (controlled).

**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. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
Major Code 03

Major Code 03 – Input operation completed successfully, but no data received.

Description: The input operation issued by your program completed successfully, but no data was received.

Action: Examine the minor return code and continue with the next operation.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 0300  | Description: On a successful input operation, your program received a turnaround indication without any data. The session is still active.  
Action: Issue an input or 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: On a successful input operation, your program received a fail indication without any data. Either the other program has sent a fail function, or the system has detected a break condition.  
Action: Issue an input operation to receive the reason for the fail from the other program. |
| 0303  | Description: On a successful input operation, your program received an end-of-group indication without any data.  
Action: Issue another input operation. |
| 0308  | Description: On a successful input operation, your program received a detach indication without any data. The communications transaction with the other program has ended. If you specified the DDS keyword RCVDETACH, the receive-detach indicator is also set on.  
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 other 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) |
| 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)

**0314 Description:** On a successful input operation, your program received a turn-around indication without any data. In addition, the other 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 other 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.

**0317 Description:** On a successful input operation, your program received an end-of-group indication without any data. In addition, the other 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 other 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 other 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 other 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.

---

**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 to the remote system. You can attempt to send the data later.

**Action:** Issue an input operation to receive the data.
**Code** | **Description/Action**
--- | ---
**0402** | **Description:** Your program was sending data when a fail indication was received. Your program is now in receive state.  
**Action:** Issue an input operation.  
**Messages:**  
CPF4806 (Notify)

**0412** | **Description:** An output exception occurred because your program attempted to send data when it should be receiving data that was sent by the other program. The data from your output operation was not sent. Your program can attempt to send the data later.  
**Action:** Issue an input operation to receive the data.  
**Note:** If your program issues another output operation before an input operation, your program receives a return code of 831C.  
**Messages:**  
CPF4750 (Notify)  
CPF5076 (Notify)

**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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 3401 | **Description**: The input operation issued by your program was not successful because the length of the data record sent by the other system was longer than the length specified for your program's input buffer. The length of the data record received from the other system, if available, is in the actual-record-length field in the I/O feedback area.  
**Action**: Issue another input operation if your program can specify a record size large enough to receive the data, plus any indicators for a file without a separate indicator area. Otherwise, you should close the file, end your program, correct the record size, then run your program again.  
**Messages**:  
CPF4768 (Notify) |

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 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) |

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 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 other 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) |
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.

---

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 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:**  
CPF4170 (Escape)  
CPF4510 (Escape)  
CPF4566 (Escape)  
CPF5257 (Escape) |

| 8082  | **Description:** The operation attempted by your program was not successful because the device supporting intrasystem communications between your program and the other program 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}

80EB Description: The open operation attempted by your program was not successful due to one of the following:

- Your program used an option of update or delete to open the file, but that option is not supported by the program device.
- Your program requested both blocked data and user buffers on an open option, but these formats cannot be selected together.
- Your program tried to open a source file, but the file was not created as a source file.
- There is a mismatch on the INDARA keyword between your program and the ICF file as to whether or not a separate indicator area should be used.
- The file was originally opened as a shared file; however, no program devices were ever acquired for the file before your program attempted the current open operation.

Action: After performing one of the following actions, your program can try the open operation again:

- If the update and delete options are not supported for the program device, use an option of input, or 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 because there is a record format level mismatch between your program and the ICF file.

Action: Close the file. 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.

Messages:

CPF4131 {Escape}

80EF Description: Your program attempted an open operation on a file or library for which the user is not authorized.

Action: Close the file. Either change the file or library name on the open operation, or obtain authority for the file or library from your security officer. Then issue the open operation again.

Messages:

CPF4104 {Escape}
80F8  **Description:** The open operation attempted by your program was not successful because one of the following occurred:

- The file is already open.
- The file is marked in error on a previous return code.

**Action:**

- If the file is already open, close the file and end your program. Remove the duplicate open operation from your program, then issue the open operation again.
- If the file is marked in error, your program can check the job log to see what errors occurred previously, then take the appropriate recovery action for those errors.

**Messages:**

CPF4132 (Escape)
CPF5129 (Escape)

---

**Major Code 81**

**Major Code 81 – Permanent session error (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 initiated the session, you can:

- Correct the problem and establish the session again. If the operation is still not successful, your program should end the session.
- Continue processing without the session.
- End.

If your session was initiated by a program start request from the other program, you can:

- Continue processing without the session.
- End.

**Note:** If the session is started again, it starts from the beginning, not at the point where the session error occurred.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>8140</td>
<td><strong>Description:</strong> 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. <strong>Action:</strong> 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. <strong>Messages:</strong> CPF5104 (Escape)</td>
</tr>
<tr>
<td>81E9</td>
<td><strong>Description:</strong> 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</td>
</tr>
</tbody>
</table>
keyword, so there was no default record format to use. The session has ended.

**Action:** Verify that the data sent by the other program was correct. If the data was not correct, change the other 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. If the operation is still not successful, your program should end.
- Continue processing without the session.
- End.

If your session was initiated by a program start request from the other program, you can:

- Correct the problem and attempt to connect to the requesting program device again. If the operation is still not successful, your program should end.
- Continue processing without the session.
- End.

Several of the minor codes indicate that an error condition must be corrected by changing a value in the communications configuration or in the file.

- To change a parameter value in the communications configuration, vary the configuration off, make the change to the configuration description, then vary the configuration on.
- To change a parameter value in the file, use the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

**Note:** When a parameter can be specified both in the ADDICFDEVE or OVRICFDEVE command and in the configuration, the value in the ADDICFDEVE or OVRICFDEVE command overrides the value specified in the configuration (for your program only). Therefore, in some cases, you may choose to make a change with the ADDICFDEVE or OVRICFDEVE command rather than in the configuration.

If no changes are needed in your file or in the configuration (and depending on what the return code description says):

- If the attempted operation was an acquire, issue the acquire operation again.
- If the attempted operation was an open, close the file and issue the open operation again.
Code  Description/Action
8209  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 (PWRDWN) 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, CHGCIFDEVE, 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, CHGCIFDEVE, or OVRICFDEVE command for 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, CHGCIFDEVE, 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:
- CPF4168 (Escape)
- CPF4370 (Escape)
- CPF4375 (Escape)
- CPF5317 (Escape)
- CPF5318 (Escape)

8282  Description: The open or acquire operation attempted by your program was not successful because the device supporting intrasystem communications between your program and the other program 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)
CPF5269 (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 CPI Communications requesting conversation is already allocated.
- 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 other program must send a program start request without a detach function.
- If your program released its *REQUESTER device, correct the error that caused your program to release its *REQUESTER device before trying to acquire it.
- 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 other 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 specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command does not match any remote location configured on the system. 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 RMTLOCNMAE 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:
- CPF4304 (Escape)
- CPF5355 (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)

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 RMTLOCNMAE 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)
82EF  Description: Your program attempted an acquire operation, or an open operation that implicitly acquires a session, to a device that the user is not authorized to, or that is in service mode. The session was not started.

Action: If the operation was an acquire, correct the problem and issue the acquire again. If the operation was an open, close the file, correct the problem, then issue the open operation again. To correct an authority error, obtain authority for the device from your security officer or device owner. If the device is in service mode, wait until machine service function (MSF) is no longer using the device before issuing the operation again.

Messages:
- CPF4104 (Escape)
- CPF4186 (Escape)
- CPF5278 (Escape)
- CPF5279 (Escape)

82F4  Description: The open or acquire operation attempted by your program was not successful because the open operation for input only is valid only for a requesting program device.

Action: End your program, correct the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, then run your program again.

Messages:
- CPF4322 (Escape)
- CPF5539 (Escape)
Major Code 83

**Major Code 83** – Session error occurred (the error is recoverable).

**Description:** A session error occurred, but the session may still be active. Recovery within your program might be possible.

**Action:** You can perform the following general actions for all 83xx return codes. Specific actions are given in each minor code description.

- Correct the problem and continue processing with the session. If the error occurred because of a resource failure on the system, 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 system operator that a change is required to correct the error received.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>830B</td>
<td><strong>Description:</strong> 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. <strong>Action:</strong> 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. <strong>Messages:</strong> CPF4079 (Diagnostic) CPF4739 (Status) CPF5067 (Escape) CPF5068 (Escape) CPF5070 (Escape)</td>
</tr>
<tr>
<td>8319</td>
<td><strong>Description:</strong> The other program sent a negative-response with sense data. <strong>Action:</strong> Issue an input operation to receive the sense data. <strong>Messages:</strong> CPF4773 (Notify)</td>
</tr>
</tbody>
</table>
831A Description: One of the following occurred:
- The evoke function attempted by your program was not successful.
- The other program issued an end-of-session function.
- The other program ended abnormally.

Action: Your program can issue an end-of-session function, issue a different evoke function, or end.

Messages:
- CPF4805 (Notify)
- CPF4808 (Notify)

831B Description: Your program tried to specify invalid sense data on a negative-response function, or it tried to send a negative-response that has already been sent to the current chain. The data was not sent.

Action: Correct your program so that it does not issue the same negative-response more than once, and that it sends valid sense data on a negative-response function. Valid sense data must be either 0 or 8 bytes long. To send 8 bytes, the first four bytes must be 0000, 08xx, or 10xx, and the remaining four bytes must be in the ranges 0-9, A-F, or a-f. If your program chooses to send a negative-response without sense data, intrasystem communications automatically sends 08110000 to the other program.

Messages:
- CPF4774 (Notify)

831C Description: Your program's previous output operation received a return code of 0412, indicating that your program must receive information sent by the other program; however, your program did not handle the return code correctly. The current output operation was not successful because your program should have issued an input operation to receive the information already sent by the other program.

Action: Issue an input operation to receive the previous information.

Messages:
- CPF4934 (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 intrasystem communications.
- Your program requested a combination of operations or keywords that was not valid, such as a combined write-then-read operation with the invite function specified.
- Your program issued an input operation, or an output operation with the invite or allow-write function, for a file that was opened for output only.
- Your program issued an output operation for a file that was opened for input only.
- Your program issued a close operation with a temporary close option.

Action: Your program can try a different operation, issue a release operation or end-of-session function, or end. Correct the error in your program before trying to communicate with the other 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:
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)

8322 Description: Your program tried to issue a negative-response or a request-to-write function. These functions are only valid while your program is in receive state.

Action: Your program can issue an output operation to continue sending data, issue an input operation to begin receiving data, issue an end-of-session function to continue local processing, or end. Correct the error that caused your program to attempt the operation that was not valid.

Messages:

CPF4703 (Notify)
CPF4775 (Notify)

8323 Description: Your program attempted to issue a cancel function when data was received for your program. The cancel function is only valid in send state.

Action: Your program can issue an input operation to continue receiving data, issue an end-of-session function, or end. Correct the error that caused your program to attempt the invalid operation.

Messages:

CPF4776 (Notify)
8326 **Description:** Your program attempted to issue a negative-response function or a cancel function to cancel a group of records when no records were previously sent to start a group. The cancel function is only valid within a chain; it is not valid preceding a chain or between chains. The session is still active.

**Action:** Correct the error that caused your program to attempt the invalid operation.

**Messages:**
- CPF4779 (Notify)
- CPF4810 (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 wants 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:**
- CPF5098 (Notify)

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 that was not valid, correct your program.

**Messages:**
- CPF5099 (Notify)

832A **Description:** Both your program and the other program were attempting to receive data at the same time.

**Action:** The other program is waiting to receive data from your program. Issue an output operation. If a coding error in your program caused the error, correct your program.

**Messages:**
- CPF4807 (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, or issue a cancel-invite function to cancel the invite function; then try the release operation again. 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, a negative-response indication, a cancel reply, 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 or release operation 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 or release operation. Correct the error that caused your program to issue an evoke function during an active transaction; then run your program again.

**Messages:**

CPF5099 (Notify)

**8330 Description:** On a successful input operation, your program received a cancel function with a turnaround indication. The other program has canceled the group of records it was sending and is now ready to receive data from your program. The session is still active.

**Action:** Normally, your program should discard the canceled data it received from the other program, as the data may be in error. Your program can then issue an output operation.

**Messages:**

CPF4782 (Notify)

**8331 Description:** On a successful input operation, your program received a cancel function without a turnaround indication. The other program has canceled the group of records it was sending, but it is still in send state, and your program is still in receive state. The session is still active.

**Action:** Normally, your program should discard the canceled data it received from the other program, as the data may be in error. Your program should then issue another input operation.

**Messages:**

CPF4783 (Notify)

**8334 Description:** The evoke function attempted by your program was not valid. A program name must be specified on the evoke function.

**Action:** Correct your program so that it issues the evoke correctly, then try the operation again.

**Messages:**

CPF4804 (Notify)

**83CD Description:** The input operation issued by your program was not successful because your program attempted a confirm function for a transaction that was started with a synchronization level of *NONE.

**Action:** Issue an end-of-session function and change your program to start the transaction with a synchronization level of *CONFIRM.

**Messages:**

CPF5016 (Notify)

**83D6 Description:** The RSPCONFIRM function issued by your program was not valid because the other program did not request confirmation, or because the current transaction was started with a synchronization level of *NONE.
Action: If the other 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)

83E8  Description: Your program attempted to issue a cancel-invite function to a session that was not invited. One of the following may have occurred:
- The invite function was implicitly canceled earlier in your program by a valid output operation.
- The invite function was satisfied earlier in your program by a valid input operation.
- Your program had already canceled the invite function, then tried to cancel it again.
- Your program never invited the session.

The session is still active.

Action: Your program can issue an input or output operation, issue an end-of-session function, continue local processing, or end. However, you should correct the error that caused your program to attempt the cancel-invite to a session that was not invited.

Messages:

CPF4763  (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)

Failed Program Start Requests

Message CPF1269 is sent to the system operator message queue when the local system rejects an incoming program start request. You can use the message information to determine why the program start request was rejected.

The CPF1269 message contains two reason codes. One of the reason codes can be zero, which can be ignored. If only one nonzero reason code is received, that reason code represents the reason the program start request was rejected. If you are running in the System/36 environment on your AS/400 system, there can be two nonzero reason codes. These two reason codes occur when the OS/400 program cannot determine whether the program start request was to start a job in the System/36 environment or by the OS/400 program. One reason code explains why the program start request was rejected in the System/36 environment and the other explains why the program start request was rejected by the OS/400 program. Whenever you receive two reason codes, you should determine which environment the job was to run in and correct the problem for that environment.
Figure B-2 on page B-25 describes reason codes for failed program start requests.

<table>
<thead>
<tr>
<th>Reason Code</th>
<th>Reason Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>Program start request received to a device that is not allocated to an active subsystem.</td>
</tr>
<tr>
<td>402</td>
<td>Requested device is currently being held by a Hold Communications Device (HLDCMNDEV) command.</td>
</tr>
<tr>
<td>403</td>
<td>User profile is not accessible.</td>
</tr>
<tr>
<td>404</td>
<td>Job description is not accessible.</td>
</tr>
<tr>
<td>405</td>
<td>Output queue is not accessible.</td>
</tr>
<tr>
<td>406</td>
<td>Maximum number of jobs defined by subsystem description are already active.</td>
</tr>
<tr>
<td>407</td>
<td>Maximum number of jobs defined by communications entry are already active.</td>
</tr>
<tr>
<td>408</td>
<td>Maximum number of jobs defined by routing entry are already active.</td>
</tr>
<tr>
<td>409</td>
<td>Library on library list is exclusively in use by another job.</td>
</tr>
<tr>
<td>410</td>
<td>Group profile cannot be accessed.</td>
</tr>
<tr>
<td>411</td>
<td>Insufficient storage in machine pool to start job.</td>
</tr>
<tr>
<td>412</td>
<td>System value not accessible.</td>
</tr>
<tr>
<td>501</td>
<td>Job description was not found.</td>
</tr>
<tr>
<td>502</td>
<td>Output queue was not found.</td>
</tr>
<tr>
<td>503</td>
<td>Class was not found.</td>
</tr>
<tr>
<td>504</td>
<td>Library on initial library list was not found.</td>
</tr>
<tr>
<td>505</td>
<td>Job description or job description library is damaged.</td>
</tr>
<tr>
<td>506</td>
<td>Library on library list is destroyed.</td>
</tr>
<tr>
<td>507</td>
<td>Duplicate libraries were found on library list.</td>
</tr>
<tr>
<td>508</td>
<td>Storage-pool defined size is zero.</td>
</tr>
<tr>
<td>602</td>
<td>Transaction program-name value is reserved but not supported.</td>
</tr>
<tr>
<td>604</td>
<td>Matching routing entry was not found.</td>
</tr>
<tr>
<td>605</td>
<td>Program was not found.</td>
</tr>
<tr>
<td>704</td>
<td>Password is not valid.</td>
</tr>
<tr>
<td>705</td>
<td>User is not authorized to device.</td>
</tr>
<tr>
<td>706</td>
<td>User is not authorized to subsystem description.</td>
</tr>
<tr>
<td>707</td>
<td>User is not authorized to job description.</td>
</tr>
<tr>
<td>708</td>
<td>User is not authorized to output queue.</td>
</tr>
<tr>
<td>709</td>
<td>User is not authorized to program.</td>
</tr>
<tr>
<td>710</td>
<td>User is not authorized to class.</td>
</tr>
<tr>
<td>711</td>
<td>User is not authorized to library on library list.</td>
</tr>
<tr>
<td>712</td>
<td>User is not authorized to group profile.</td>
</tr>
<tr>
<td>713</td>
<td>User ID is not valid.</td>
</tr>
<tr>
<td>714</td>
<td>Default user profile is not valid.</td>
</tr>
<tr>
<td>715</td>
<td>Neither password nor user ID was provided, and no default user profile was specified in the communications entry.</td>
</tr>
<tr>
<td>718</td>
<td>No user ID.</td>
</tr>
<tr>
<td>722</td>
<td>A user ID was received but a password was not sent.</td>
</tr>
<tr>
<td>723</td>
<td>No password was associated with the user ID.</td>
</tr>
<tr>
<td>725</td>
<td>User ID does not follow naming convention.</td>
</tr>
<tr>
<td>726</td>
<td>User profile is disabled.</td>
</tr>
<tr>
<td>801</td>
<td>Program initialization parameters are present but not allowed.</td>
</tr>
<tr>
<td>802</td>
<td>Program initialization parameter exceeds 2000 bytes.</td>
</tr>
<tr>
<td>803</td>
<td>Subsystem is ending.</td>
</tr>
<tr>
<td>804</td>
<td>Prestart job is inactive or is ending.</td>
</tr>
<tr>
<td>805</td>
<td>WAIT(NO) was specified on the prestart job entry and no prestart job was available.</td>
</tr>
<tr>
<td>806</td>
<td>The maximum number of prestart jobs that can be active on a prestart job entry was exceeded.</td>
</tr>
<tr>
<td>807</td>
<td>Prestart job ended when a program start request was being received.</td>
</tr>
<tr>
<td>901</td>
<td>Program initialization parameters are not valid.</td>
</tr>
<tr>
<td>902</td>
<td>Number of parameters for program not valid.</td>
</tr>
<tr>
<td>903</td>
<td>Program initialization parameters required but not present.</td>
</tr>
<tr>
<td>Reason Code</td>
<td>Reason Description</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>1001</td>
<td>System logic error. Function check or unexpected return code encountered.</td>
</tr>
<tr>
<td>1002</td>
<td>System logic error. Function check or unexpected return code encountered while receiving program initialization parameters.</td>
</tr>
<tr>
<td>1501</td>
<td>Character in procedure name not valid.</td>
</tr>
<tr>
<td>1502</td>
<td>Procedure not found.</td>
</tr>
<tr>
<td>1503</td>
<td>System/36 environment library not found.</td>
</tr>
<tr>
<td>1504</td>
<td>Library QSSP not found.</td>
</tr>
<tr>
<td>1505</td>
<td>File QS36PRC not found in library QSSP.</td>
</tr>
<tr>
<td>1506</td>
<td>Procedure or library name is greater than 8 characters.</td>
</tr>
<tr>
<td>1507</td>
<td>Current library not found.</td>
</tr>
<tr>
<td>1508</td>
<td>Not authorized to current library.</td>
</tr>
<tr>
<td>1509</td>
<td>Not authorized to QS36PRC in current library.</td>
</tr>
<tr>
<td>1510</td>
<td>Not authorized to procedure in current library.</td>
</tr>
<tr>
<td>1511</td>
<td>Not authorized to System/36 environment library.</td>
</tr>
<tr>
<td>1512</td>
<td>Not authorized to file QS36PRC in System/36 environment library.</td>
</tr>
<tr>
<td>1513</td>
<td>Not authorized to procedure in System/36 environment library.</td>
</tr>
<tr>
<td>1514</td>
<td>Not authorized in library QSSP.</td>
</tr>
<tr>
<td>1515</td>
<td>Not authorized to file QS36PRC in QSSP.</td>
</tr>
<tr>
<td>1516</td>
<td>Not authorized to procedure in QS36PRC in QSSP.</td>
</tr>
<tr>
<td>1517</td>
<td>Unexpected return code from System/36 environment support.</td>
</tr>
<tr>
<td>1518</td>
<td>Problem phase program not found in QSSP.</td>
</tr>
<tr>
<td>1519</td>
<td>Not authorized to problem phase program in QSSP.</td>
</tr>
<tr>
<td>1520</td>
<td>Maximum number of target programs started (100 per System/36 environment).</td>
</tr>
<tr>
<td>2501</td>
<td>System logic error. Function check or unexpected return code encountered while processing a program start request.</td>
</tr>
<tr>
<td>2502</td>
<td>Temporarily unable to allocate needed resources for a program start request.</td>
</tr>
<tr>
<td>2503</td>
<td>No subsystem accepting program start requests for this device.</td>
</tr>
</tbody>
</table>
Appendix C. Using Intrasystem Communications to Test Applications

This appendix discusses the differences between intrasystem communications and other communications types, such as advanced program-to-program communications (APPC), binary synchronous communications equivalence link (BSCEL), Systems Network Architecture Upline Facility (SNUF), asynchronous, retail, and finance communications, in sending and receiving data. There may also be differences in the way the communications types support the start of sessions and transactions, online messages, record length, and return codes.

These differences should be noted, especially if you use intrasystem communications to test new application programs to be run using other communications types.

If your program expects to receive certain return codes and messages, these codes and messages may not be the same for intrasystem communications as they are for another communications type. You may only be able to observe a range of messages, or you may have to refer to the return codes section for a specific communications type to determine the differences for each code.

Using Intrasystem Communications for Advanced Program-to-Program Communications

Advanced program-to-program communications (APPC) allows programs on an AS/400 system to communicate with programs on other systems having compatible communications support. It also provides the capability for two programs to communicate with each other while running on the same AS/400 system. This capability is enabled when LINKTYPE(*LOCAL) is specified when the APPC controller description is created. We recommend this method to test programs which you have written to use APPC. APPC is the AS/400 implementation of the SNA LU session type 6.2 architecture. The following considerations apply when you use intrasystem communications to test programs to be run using APPC.

Confirm Function

When your program uses the confirm function, intrasystem communications sends a positive response to a confirm request if the user does a valid read or write operation. However, APPC requires the user to use the RSPCONFIRM DDS keyword to send a positive response.

Intrasystem communications also supports the confirm function with non-APPC functions, such as the end-of-group function. This results in more return codes for intrasystem communications.

Conversation Types

APPC supports both basic and mapped conversation types; intrasystem communications only supports the equivalent of a mapped conversation. Programs written for either basic or mapped conversations can run using intrasystem communications. However, intrasystem communications does no checking to ensure the two communicating programs are both using the same conversation type.

Evoke Function

When your program uses the evoke function to start another program and you specify *USER for the user ID on the SECURITY keyword, intrasystem communications always passes the user ID on the program start request. However, APPC only passes the user ID if the remote system accepts a user ID that has already been verified.

Fail Function

If your program receives a fail indication, intrasystem communications issues a 0302 return code when your program is in the receive state, and a 0402 return code when your program is in the send state. However, APPC issues 83C7 through 83CC return codes when a fail indication is received in the send or receive state.

Force-Data Function

Intrasystem communications does not buffer data, APPC does. If you use APPC, and your program issues a write operation without specifying a function that forces the data to be sent (for example, an invite function, force-data function, or read operation), the data is buffered so it can be sent later.

Output Operations

If your program attempts to send data when it should be receiving data, an output exception occurs and intrasystem communications sends your program a 0412 return code. If your program issues another output operation, it receives a 831C return code. APPC continues to send the 0412 return code.

Record Length

On input operations, if the length of the data record sent by the other program is greater than the length of your program input buffer, intrasystem communications returns a 3401 return code, and your program can issue another read operation if it can specify a record size large enough to receive the entire record. APPC truncates the data to fit in your program's input buffer, and returns a 3431 return code with the data; the data that was truncated is lost.

Variable Buffer Management (VARBUFMGT)

APPC supports the variable buffer management (VARBUFMGT) DDS keyword, whereas intrasystem communications does not.
Using Intrasystem Communications for Asynchronous Communications

Asynchronous communications is a method of communications that allows an exchange of data with a remote device or system, using either a start-stop line or an X.25 line. The following considerations apply when you use intrasystem communications to test programs to be run using asynchronous communications.

Detach Function
Intrasystem communications requires that your program issue a detach function to end a transaction before ending the session. Asynchronous communications does not support the detach function, and does not require a transaction to be ended before ending the session.

Evoke Function
Intrasystem communications requires that your program issue an evoke function as the first operation after an acquire operation, whereas asynchronous communications does not. Therefore, your first operation following an acquire operation must be an evoke function when you use intrasystem communications to test an application program which is to be run using asynchronous communications.

Fail Function
For both intrasystem and asynchronous communications, your program issues the fail function to indicate that it has detected an error in the data while it was sending or receiving. However, whereas asynchronous communications always sends a 0302 return code, intrasystem communications sends a 0302 return code when your program is in receive state, and a 0402 return code when your program is in send state. Also, asynchronous communications discards all data waiting to be received by your application whenever a fail indication is sent or received.

Function-Management-Header Function
If your program uses the function-management-header function, intrasystem communications sends the function-management-header data to the other program. If your program uses asynchronous communications and issues a write function-management-header function, it affects data translation, changes certain characteristics of data on an asynchronous communications line, or sends packet assembler/disassembler (PAD) messages.

Number of Sessions
Intrasystem communications allows multiple sessions per device; asynchronous communications allows only one session per device.

Read or Write Operations
Asynchronous communications allows your program to issue read and write operations in any order. Intrasystem communications normally requires that your application program be in the send state to issue output operations, and in the receive state to issue input operations; however, if your program is in the send state, you may issue a read operation.

Translation
Asynchronous communications supports translation of data from EBCDIC to ASCII, whereas intrasystem communications does not.

Using Intrasystem Communications with Binary Synchronous Communications Equivalence Link

Binary synchronous communications (BSC) is a data communications line protocol that uses a standard set of transmission control characters and control character sequences to send binary-coded data over a communications line. The ICF support on the AS/400 system that provides binary synchronous communications with another AS/400 system is referred to as binary synchronous communications equivalence link (BSCEL) support. The following considerations apply when you use intrasystem communications to test programs to be run using binary synchronous communications equivalence link (BSCEL).

Detach Function
Intrasystem communications requires that your program use a detach function to end a transaction, and the other program receives a minor return code of 08 indicating that the detach function was sent. If your program uses BSCEL and specifies RMTBSCEL(*NO), BSCEL treats a detach function as if it were an end-of-group function, and the receiving program would never receive a 08 minor return code.

End-of-Group Function
Intrasystem communications issues a return code of 0003 or 0303 to the other program when your program issues an end-of-group function (the '03' minor code indicates an end of group). BSCEL issues either a 0300 or 0301 return code to the receiving program, depending on the value specified for the GRPSEP parameter in the device description or on the ADDICFDEVE, the CHGICFDEVE, or the OVRICFDEVE command.

Evoke Function
Intrasystem communications requires that a source program issue an evoke function as the first operation after an acquire operation. However, if your program uses BSCEL and you specify RMTBSCEL(*NO) on a program device entry command, on the CRTDEVBSC command, or the CHGDEVBSC command, the evoke function is optional, and the first input or output operation from your program starts the transaction.

If your program uses intrasystem communications and the evoke function fails, a notify message is sent to your program with a reason code indicating why it failed. If your program uses BSCEL and you specify RMTBSCEL(*YES) and an evoke function fails, both a notify message and an online message are sent to
your program and your program must issue a read operation to receive the online message.

Fail Function
When intrasystem communications support receives a fail function, a return code of 0402 or 0302 is returned to your program, and you may correct the error indicated and continue sending data. When BSCEL receives a fail function (an end-of-transmission, or EOT, indication), a return code of 8197 or 8198 is returned to your program, and the session is ended.

Number of Sessions
Intrasystem communications allows multiple sessions per device; BSCEL allows only one session per device.

Online Messages
Intrasystem communications does not send or receive any online messages; BSCEL does support online messages.

Program Start Requests
When using BSCEL, a source program that specifies RMTBSCEL(*NO) for the communications session can send data in the proper format for a program start request with the program's first output operation. When using intrasystem communications, your source program cannot issue a program start request; the invoke function must be used to start another program.

Receiving Data
If you are using intrasystem communications, a return code of 0300 or 832A is used if both your program and the program with which you are communicating attempt to receive data at the same time. If you use BSCEL, both programs will be waiting to receive data indefinitely.

Receive-Turnaround Indication
If you use intrasystem communications, your program may receive a turnaround indication on the same read operation for which your program receives data. BSCEL sends the turnaround indication as a separate transmission after the data record is sent. Therefore, you may need to issue an additional read operation for BSCEL to receive the turnaround indication.

Record Blocking
Intrasystem communications does not support record blocking (that is, you cannot specify the BLOCK parameter on the program device entry commands); BSCEL does support record blocking.

Record Length
Intrasystem communications supports a maximum record length of 32767 bytes; BSCEL supports a maximum record length of 8192 bytes.

Using Intrasystem Communications for Finance Communications

Finance communications allows programs on an AS/400 system to communicate with programs using the SNA LU session type 0 protocol. The following considerations apply when you use intrasystem communications to test programs to be run using finance communications.

Allow-Write and Request-to-Write Functions
Both intrasystem and finance communications require that your program either send or receive at any given time. However, whereas intrasystem communications uses the allow-write and request-to-write functions as a way of determining which program should send or receive, finance communications does not. If you use finance communications, and neither your program nor the controller has sent a group of records, a contention state exists, in which either program may attempt to send. If both the local and the controller program send at the same time, the controller is designated the sender, and can send a negative-response indication to your program. When writing programs that use finance communications, you need to be aware of these contention error conditions.

Confirm, End-of-Group, Invite, or Read Functions Specified on Write Operations with Data
Intrasystem communications allows your program to specify a confirm function on write operations with data; finance communications does not. However, finance communications supports a function similar to the confirm function. When your program issues a write operation with data, and the end-of-group function is also specified, the data is sent to the finance controller, and the write operation does not complete until a response is received from the controller.

If you specify invite or read functions on a write operation with data, however, the data is sent to the finance controller as if an end-of-group function was specified, but no response from the finance controller is required.

Note: Data sent to a 3694 finance controller never requires a response.

Force-Data Function
Intrasystem communications does not buffer data; finance communications does. If you use finance communications, and your program issues a write operation and this is the first record in a group of records, the data is sent immediately. However, if your program sends subsequent records without specifying a function that closes the group of records (for example, end-of-group or invite functions or a read operation), the data may be buffered to be sent at a later time. Your program can use the force-data function to ensure that data is sent when the write operation is issued.
Number of Sessions
Intrasystem communications allows multiple sessions per device; finance communications allows only one session per device.

Read Operations
Intrasystem communications returns 0000 and 0001 return codes on read operations, finance communications does not. If you use finance communications; your program must receive an entire group of records. Therefore, your program can only receive the following return codes if the read operation is successful: 0003 (the last record in a group of records has been received) or 0007 (a group of records was received with a function management header as the first record).

Sense Data
Finance communications returns sense data to your program in an I/O feedback area that is accessible to your program. Sense data is returned for any operation that fails with an 8319 or 831A return code. However, intrasystem communications requires that the user issue an input operation to receive the sense data.

Write Operations
If you use intrasystem communications, your program receives an error indication on a write operation if the error indication is received before the write operation is issued. When finance communications receives an error indication before your program issues a write operation, your program either receives the error indication on the write operation, or is required to issue an input operation to receive the error indication if the session is invited.

Using Intrasystem Communications for Retail Communications
Retail communications allows programs on an AS/400 system to communicate with programs using SNA LU session type 0 protocol. The following considerations apply when you use intrasystem communications to test programs to be run using retail communications.

Sending and Receiving Data
Intrasystem communications is half-duplex, that is, your program can send or receive data, but cannot do both at the same time. Retail communications allows you to acquire sessions with retail controllers using a Systems Network Architecture (SNA) bind command that specifies a duplex protocol, that is, you can send and receive data at the same time. Therefore, when writing programs that use retail communications, you should note that it does not support the usual rules relating to when your program can send or receive data.

Note: Due to this major difference between intrasystem and retail communications, using intrasystem communications may not be the most effective way to test programs that use retail communications.

Confirm, End-of-Group, Invite, or Read Functions Specified on Write Operations with Data
Intrasystem communications allows your program to specify a confirm function on write operations with data; retail communications does not. However, retail communications supports a function similar to the confirm function. When your program issues a write operation with data, and the end-of-group function is also specified, the data is sent to the retail controller, and the write operation does not complete until a response is received from the controller.

If you specify invite or read functions on a write operation with data, however, the data is sent to the retail controller as if an end-of-group function was specified, but no response from the retail controller is required.

Detach Function
Intrasystem communications allows your program to send data when using the detach function, but retail communications does not. In addition, retail communications requires that any partially sent or partially received group of records be closed before a detach function is allowed. If any data or error indications have been received from the retail controller but have not yet been received by your program, the detach function fails, and the return code 8322 is returned to your program.

Evoke Function
When you use intrasystem communications and issue an evoke function, you must specify the program name. You may also use, for example, the SECURITY keyword and program initialization parameters, and specify read operations or functions such as the invite and function-management-header functions. If the retail controller program specifies program initialization parameters or security information on the evoke function, retail communications ignores this information. If the function-management-header function is specified on an evoke function, retail communications issues an 831E return code.

Force-Data Function
Retail communications buffers data; intrasystem communications does not. If your program issues a write operation without specifying a function that closes the group of records (for example, an end-of-group, force-data, or invite function or a read operation), the data may be buffered so that it can be sent at a later time. Specifying any of these functions ensures that all the data is sent.

Invite Function
If your program sends a group of records and then issues an invite function, intrasystem communications closes the group of records that is being sent. However, retail communications does not; the session is simply invited.
Note: Retail communications does close a group of records that your program is sending, however, when you issue an invite function and it is specified on a write operation with data.

Number of Sessions
Intrasystem communications allows multiple sessions per device; retail communications allows only one session per device.

Read Operations
Intrasystem communications returns a 0001 return code on a read operation; retail communications does not. A retail application program can receive one of the following return codes: 0000 (one record was received in a group of records); 0003 (the last record was received or the only record of a group of records was received); 0005 (the first record of a group of records was received with a function-management-header indication); or 0007 (a group of records was received with function-management-header data as the first record). In addition, if you partially send a group of records on a write operation, and then issue a read operation, intrasystem communications closes the group of records. However, retail communications only closes a group of records on a read operation if it is specified on a write operation with data.

Sense Data
Retail communications returns sense data to your program in an I/O feedback area that is accessible to your program. Sense data is returned for any operation that fails with an 8319 or 831A return code. However, intrasystem communications requires that the user issue an input operation to receive the sense data.

Write Operations
Intrasystem communications requires your program to receive data when data is available to be received. You cannot also send data while you are receiving. However, retail communications allows your program to issue write operations at any time. The only case in which this is not true is when you receive an error indication from the remote program.

Using Intrasystem Communications for Systems Network Architecture Upline Facility

The SNA upline facility (SNUF) is the communications support that allows the AS/400 system to communicate with CICS/VS and IMS/VS application programs on a host system. The following considerations apply when you use intrasystem communications to test programs to be run using Systems Network Architecture upline facility (SNUF).

End-of-Group Function
Intrasystem communications issues a return code of 0003 or 0303 to indicate the end of a group of records. SNUF issues a major return code of 00 with minor return codes 01 or 03, or a major return code 03 with minor return codes 01 or 03, depending on the value specified for the batch parameter on either the Create Device Description (SNUF) (CRTDEVSNUF) command or on the ADDICFDEVE or OVRICFDEVE command.

Evoke Function
Intrasystem communications requires your program to issue an evoke function after acquiring a session to start the target program; SNUF does not.

Fail Function
Intrasystem communications can send and receive fail indications in addition to cancel and negative-response indications. If your program receives a fail function while in the receive state, intrasystem communications returns a 0302 return code. If your program receives a fail function while in send state, intrasystem communications returns a 0402 return code. SNUF cannot receive a fail indication from the host system, but can receive a cancel or negative-response indication, depending on the state of the transaction. If your program issues a fail function while in receive state, SNUF sends a negative-response indication to the host system. If your program issues a fail function while in the send state, SNUF sends a cancel indication to the host system.

Function-Management-Header Data
If a function-management-header indication is received with data, intrasystem communications inserts the characters FMH before the data that is received. SNUF does not insert these characters; instead, the return code indicates that the function-management-header indication is being received.

Number of Sessions
Intrasystem communications allows multiple sessions per device, SNUF allows only one session per device.

System Messages
Intrasystem communications does not send or receive system messages. However, SNUF checks the received data for system messages, and if they are received, SNUF issues a major return code 00 with the following minor return codes: 20, 21, 23, 25, 27, 28, 30, 31, 33, 35, 37, or 38.
Appendix D. Program Examples

This appendix provides sample programs to demonstrate how intrasystem communications is used. Program examples for the ILE C/400, C Set ++, COBOL*400, and RPG*/400 programming languages are provided with explanations.

In the ILE C/400 program example, both the source and the target programs are provided to show how an application can be involved in an interactive inquiry with a single ICF session. A source program accepts inquiries from a display device and sends a request to a target program. The source program communicates with the display device through a display file, and with the target program through an ICF file.

In the COBOL/400 and RPG/400 program examples, both source and target programs are provided in an example that illustrates how an application can be involved in an interactive inquiry with two ICF sessions. A source program accepts inquiries from a display device and sends a request to one of two target programs. The source program communicates with the display device through a display file, and with the two target programs through a single ICF file. The purpose of the example is to show how a source program can communicate with two sessions from a single ICF file. From the viewpoint of each of the two target programs, the requester is the only session. Therefore, the target programs do not require any unique logic to support the two-session source.

For both source and target programs, the DDS source for the ICF file, program listings, and an explanation of the programs are provided.

Description of the Single-Session Inquiry Program Example

The following explanation describes the transaction between a source program and a target program, and applies to the ILE C/400 programs in this appendix.

The source program is started from a display station, and both the display and the ICF files are opened. The work station is implicitly acquired when the display file opens, but because the ICF file is created with ACQPGMDEV(*NONE), no ICF devices are acquired during open processing.

The ICF program device, ICF00, is explicitly acquired by the source program using the acquire operation. After the acquire, the source program starts the target program, through a CL program, using an evoke function. The target program also explicitly acquires a program device.

The source and target programs use their respective ICF files with program device ICF00, which is defined as the default program device from the acquire operation. Both programs will have only one session active at a time.

A customer inquiry prompt is displayed on the workstation by the source program. The source program uses a write-with-invite function to send a number entered by the user to the target program, and then waits for data to be received from the target program. The target program reads the number sent by the source program, and searches a data file for a customer record based on the number received. If a record was found, it is sent to the source program; if not, a fail indication is sent. The source program ends by sending a detach request to the target program, issuing an end-of-session operation, and closing the files that were opened.

ILE C/400 Source Program for a Single-Session Inquiry

The following describes an ILE C/400 source program for a single-session inquiry.

Program Files: The ILE C/400 single-session source program uses the following files:

SRCICFF An ICF file used to send records to, and receive records from, the target program.

DSPFIL A display file used to enter requests that are to be sent to the target program.
**DDS Source:** The DDS for the ICF file (SRCICFF) is illustrated in Figure D-1.

---

**Figure D-2 (Part 1 of 2). DDS Source for a Single-Session Source Program Using DSPFIL**

The DDS source file for the display file (DSPFIL) is illustrated in Figure D-2.

---

**Figure D-2 (Part 2 of 2). DDS Source for a Single-Session Source Program Using DSPFIL**

Configuration: The following command is needed to create the intrasystem communications device associated with the ICF file:

```
CRTDEVINSTR DEVD(INTRADEV)
RMTLOCNAME(INTRARMT) ONLINE(*NO)
TEXT("THIS IS AN INTRASYSTEM DEVICE DESCRIPTION")
```

**ICF File Creation and Program Device Entry Definition:**

The following command is needed to create the ICF file:

```
CRTICFFFILE FILED(CEXAMPLES/SRCICFF)
SRCFILE(CEXAMPLES/QDDSSRC) SRCBR(SRCICFF)
ACQPGMDEV(*NONE) MAXPGMDEV(1)
TEXT("SOURCE ICF FILE FOR SINGLE SESSION PROGRAM")
```

It is not necessary to add a communications entry to the subsystem because the system automatically defines an entry for the device created above when the program is processed. However, the following command is an example of what you would use if you decided to add a communications entry:

```
ADDCMNE SBSD(QCMN) DEV(INTRADEV)
```

**Note:** Subsystem QCMN should be stopped before ADDCMNE is entered, and then started again.

The following command is needed to define the program device entry:

```
QVRICFDEV PGMDEV(ICFOO) RMTLOCNAME(INTRARMT)
FMTSLT(*PGM)
```

The following CL program could be used to run the source program:
The following explains the structure of the program example illustrated in Figure D-3 on page D-4. The ICF file used in the example is defined by the user, and uses externally described data formats (DDS). The reference numbers in the explanation below correspond to the numbers in the following program example.

All output operations to the ICF file in the example are done using the write statement with the record format name specified previously with a _Rformat function.

1. The display file descriptions (DSPFIL) are included in the program.
2. The ICF file descriptions (SRCICFF) are included in the program.
3. The routines are defined so the ILE C/400 compiler knows the type of value returned and the type of parameters passed, if any.
4. The ICF file is opened for record I/O with the separate indicator area option specified.
5. The display file is opened for record I/O with the separate indicator area option specified.
6. The separate indicator area is initialized and defined for DSPFIL. The variable dsp_indic is a 99 character array.
7. Program device ICF00 is explicitly acquired with the _Racquire function. A session is implicitly acquired for the work station when DSPFIL is opened.
8. The PGMSTR format name is specified and an evoke operation is performed with the _Rwrite function.
9. The program loops until either F3 is pressed from the work station, which sets an indicator in the display file's separate indicator area, or an error occurs in the transaction with the target program.
10. The get_cust_num() function is called to get a customer number from the user using DSPFIL.
11. The get_cust_info() function is called to send a customer number to the target program, and then to receive the customer information if it was found by the target program.
12. The get_cust_num() function gets a customer number from the user. The program displays the customer number inquiry, and reads the number.
13. The get_cust_info() function sends the customer number to the target program, and then receives the customer information sent by the target program. The customer number is copied into the output buffer of the ICF file for record format CUST defined in SRCICFF. CUST is specified on the _Rformat function, and a write is issued to the ICF file. The record format CINFO defined in SRCICFF to receive the customer information from the target is specified, and a read is issued to the ICF file.

The major return code is checked for a successful operation. If a 00 major return code is received, the customer information is displayed by calling the display_info() function, and control returns to main(). If a 00 major return code was not received, then a check is made to see if a 0302 or a 0402 return code was received, indicating that the target program issued a fail operation because it could not find customer information based on the number sent. If none of the above return codes is received, then an unexpected error has occurred and the program ends.

14. The display_info() function writes the customer information received from the target program to the work station.
15. The end_job() procedure is called when F3 is pressed to issue a detach operation to the target program, followed by an end-of-session operation to end the session.
16. The end_error() procedure is called if an error has occurred in trying to end the session. A detach is not issued since the target may have ended abnormally.
17. The send_eos() procedure issues the end-of-session operation to the ICF file.
18. The pos_resp() function checks for a 00 major return code in the display/ICF I/O feedback area.
19. The fail_rt_cd() function checks for a 0302 or a 0402 return code in the display/ICF I/O feedback area.
20. The get_access_to_fb() function accesses the display/ICF I/O feedback area by first accessing the common I/O feedback area to obtain an offset. The offset is added to the pointer to the common I/O feedback area to get access to the display/ICF I/O feedback area. The _Riobfbk function returns a pointer to the common I/O feedback area.
** **** P R O L O G ****

<table>
<thead>
<tr>
<th>Program name</th>
<th>Source member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSRCPGM</td>
<td>CSRCPGM</td>
</tr>
</tbody>
</table>

**Library name:** CEXAMPLES

**Source file:** QCSRC

**Compiler options:** +SOURCE +NOLIST +NOXREF +GEN +NOATR +NODUMP +NOOPTIMIZE +NOALWBND +NODEBUG +NOLIST +NOXREF +GEN +NOATR +NODUMP +NOOPTIMIZE +NOALWBND

**Language level options:** EXTENDED

**Source margins:**
- Left margin: 1
- Right margin: 32767

**Define name:**
- NOERROR
- ERROR
- NOEND
- END
- OFF
- ON

**Include files:**
- <stdio.h>
- <recio.h>
- <stddef.h>
- <stdlib.h>
- <string.h>
- <xxfdbk.h>

---

**Source description:**

This program reads a customer number from the display. The number is sent to the target program, ctgtpgm, which searches a data file for information about the customer. If the information is found, the data is received and printed on the display. If no information is found for the given customer number, then the user is prompted for another number.

---

**Compiler:** IBM ILE C/400 Compiler

---

**Figure D-3 (Part 1 of 6). Source Program Example — CSRCPGM**

---

D-4 Intrasystem Communications Programming V4R1
```c
main() {
    char dsp_indic[99];  /* Display separate indic area */
    
    /* Open the ICF file */
    if ((icffptr = _Ropen("CEXAMPLES/SRCICFF", "ar+ indicators=y riofb=y")) == NULL) {
        printf("ICF file failed to open.
"); 
        exit(ERROR); 
    }

    /* Open the display file */
    if ((dspfptr = _Ropen("CEXAMPLES/DSPFIL", "ar+ indicators=y riofb=y")) == NULL) {
        printf("Display file failed to open.
"); 
        _Rclose(icffptr);
        exit(ERROR); 
    }

    /* Set up separate indicator area space */
    memset(dsp_indic, OFF, 99);
    _Rindara(dspfptr, dsp_indic);

    /* Acquire a session */
    _Racquire(icffptr, "ICF/zerodot/zerodot");
    if (pos_resp() == ERROR) {
        printf("Acquire failed.
");
    } else {
        /* Evoke the other program */
        _Rformat(icffptr, "PGMSTR");
        _Rwrite(icffptr, NULL, "\n");
        if (pos_resp() == ERROR) {
            printf("Evoke failed.\n");
            end_error();
            return(ERROR); 
        }
    }

    /* Check if the evoke was successful */
    if (pos_resp() == ERROR) {
        printf("Evoke failed.\n");
        end_error();
        return(ERROR); 
    }
}
```

Figure D-3 (Part 2 of 6). Source Program Example — CSRCPGM
/* While F3 isn't pressed get the customer number */
/* and display the customer record (if received) */

while (dsp_indic??(2??) == OFF) {
/* read the customer number from the display */

/* Attempt to get customer information from the */
/* target and display the customer record if found */
if (get_cust_info(dsp_indic) == ERROR) {
  /* An error occured in the transaction */
  end_error();
  return(ERROR);
}

/* F3 was pressed, end the session and the job */
end_job();
}

/* Get a customer number from the display. */
get_cust_num(char dsp_indic??(99??)) {
  /* Put out display and get information */
  _Rformat(dspfptr, "HEADER");
  _Rwrite(dspfptr, NULL, /zerodot);
  _Rformat(dspfptr, "FOOT");
  _Rwrite(dspfptr, NULL, /zerodot);
  memset(dsp_indic, '/zerodot', 99);
  _Rreadn(dspfptr, &prompt_dsp_i, sizeof(prompt_dsp_i), __DFT);
  /* Check if F3 (end the job) was pressed */
  if (dsp_indic??(2??) == OFF) {
    return(NOEND);
  } else {
    return(END);
  }
}

/* Evoke the target program, and then send the customer number to the */
/* target. The target program should either send a customer record or */
/* a fail indication (customer not in data file). The record is dis- */
/* played with a call to display_info. */
get_cust_info(char dsp_indic??(99??))
| /* Put number in ICF file and send it to the other program */
|
| strncpy(cust_icf_o.NUMBER, prompt_dsp_i.CNUMBR, 5);
| _Rformat(icffptr, "CUST");
| _Rwrite(icffptr, &cust_icf_o, sizeof(cust_icf_o));
| /* Check if the number was sent successfully */
| printf("Unexpected error in transaction."n);
| return(ERROR);
|
| // Read the customer information */
| _Rformat(icffptr, "CINFO");
| _Rreadn(icffptr, &cinfo_icf_i, sizeof(cinfo_icf_i), __DFT);
| // Check if the record was returned, and if so print it on the */
| // display, or the fail indicator may have been received from */
| // the target program if the record wasn't found. */
| if (pos_resp() == NOERROR) {
| // Display the record, and then return to main with the value */
| // returned from display_info. */
| return(display_info(dsp_indic));
| }
| return(NOERROR);
| else {
| printf("Unexpected error in transaction."n);
| return(ERROR);
| }
|
| /* Display the customer information. */
| _Rformat(dspfptr, "HEADER");
| _Rwrite(dspfptr, NULL, __DFT);
| _Rformat(dspfptr, "FOOT");
| _Rwrite(dspfptr, NULL, __DFT);
| /* Put out display of customer information */
| _Rformat(dspfptr, "DTLSCR");
| strncpy(dtlscr_dsp_o.CUSTNO, cinfo_icf_i.CUSTNO, 5);
| strncpy(dtlscr_dsp_o.NAME, cinfo_icf_i.NAME, 20);
| strncpy(dtlscr_dsp_o.ADDR, cinfo_icf_i.ADDR, 20);
| strncpy(dtlscr_dsp_o.CITY, cinfo_icf_i.CITY, 20);
| strncpy(dtlscr_dsp_o.STATE, cinfo_icf_i.STATE, 2);
| strncpy(dtlscr_dsp_o.ZIP, cinfo_icf_i.ZIP, 5);
| strncpy(dtlscr_dsp_o.ACCBAL, cinfo_icf_i.ACCBAL, 6);
| _Rwrite(dspfptr, &dtlscr_dsp_o, sizeof(dtlscr_dsp_o));
| memset(dsp_indic, ' ', 99);
| _Rreadn(dspfptr, NULL, __DFT);

Figure D-3 (Part 4 of 6). Source Program Example — CSRCPGM
/* Check if F3 (end the job) was pressed */

if (dsp_indic??(0) == OFF)
    return(NOEND);
else
    return(END);

/***************************************************************************/

/* F3 was pressed, end the job. */

void end_job()
{
    /* Issue a detach to the target program, then end the session */
    _Rformat(icffptr, "SENDDETACH");
    _Rwrite(icffptr, NULL, /zerodot/); send_eos();
}

/***************************************************************************/

/* Error, clean up. */

void end_error()
{
    send_eos();
    _Rclose(icffptr);
    _Rclose(dspfptr);

    /***************************************************************************/

    /* Issue an end of session operation. */

    void send_eos()
    {
        _Rformat(icffptr, "ENDSESSION");
        _Rwrite(icffptr, NULL, /zerodot/);}
    
    /***************************************************************************/

    /* Check the major return code for a successful operation - 00. */

    pos_resp()
    {
        get_access_to_fb();
        if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == /zerodot/)
            return(NOERROR);
        else
            return(ERROR);
    }

    /***************************************************************************/

    /* Check the major/minor return code for a fail - 0302 or 0402. */

    fail_rt_cd()
    {
        get_access_to_fb();
        if (strncmp(dsp_icf_fdbk->major_ret_code, "03", 2) == 0 ||
            strcmp(dsp_icf_fdbk->major_ret_code, "04", 2) == 0) &
            return(NOERROR);
        else
            return(ERROR);
    }

Figure D-3 (Part 5 of 6). Source Program Example — CSRCPGM
ILE C/400 Target Program for a Single-Session Inquiry

The following describes an ILE C/400 target program for a single-session inquiry.

Program Files: The ILE C/400 single-session target program uses the following files:

TGTICFF An ICF file used to send records to, and receive records from, the source program. It is done with the file-level INDARA DDS keyword, indicating a separate indicator area.

CUSMSTP A physical file that contains customer records to be sent to the source program.

CUSMSTL A logical file used with CUSMSTP to access the customer records.

DDS Source: The DDS for the ICF file (TGTICFF) is illustrated in Figure D-4.

---

The following explains the structure of the program example illustrated in Figure D-7 on page D-11. The ICF file used in the example is defined by the user, and uses externally described data formats (DDS). The reference letters in the example below correspond to those in the following program example.

1. The database logical file descriptions (CUSMSTL) are included in the program.
2. The ICF file descriptions (TGTICFF) are included in the program.
The routines are defined so the ILE C/400 compiler knows the type of value returned and the type of parameters passed, if any.

The ICF file is opened for record I/O with the separate indicator area option specified.

The database logical file is opened for record input. If an error occurs, the ICF file is closed and the program ends.

The ICF00 program device is explicitly acquired with the _Racquire function.

The program loops until either a detach is received, or an error occurs in the transaction with the source program.

The process_data() function receives a customer number from the source program, reads the database file using the number as the key, and either returns customer data to the source program or issues a fail operation to tell the other program that customer information could not be found for the given number.

The send_eos() procedure issues an end-of-session operation to the ICF file to end the session.

The pos_resp() checks for a 00 major return code. The return code is obtained by accessing the common I/O feedback area to get a pointer to the display/ICF I/O feedback area. A pointer to the common I/O feedback area is returned from the QXXIOFBK function. The offset to the display/ICF I/O feedback area is added to the common I/O feedback area pointer to get a pointer to the display/ICF I/O feedback area.
Program name : CTGTPGM
Library name : EXAMPLES
Source file : QCSRC
Library name : EXAMPLES
Source member name : CTGTPGM
Text Description : Target C program for Intra
Compiler options : -SOURCE -NODUMP -NOSHOWUSR -NOSHOWSYS -NOSHOWUSR
Language level options : -EXTENDED
Source marginals:
Left margin : 1
Right margin : 32767
Sequence columns:
Left Column : 
Right Column : 
Define name options:
-NOFILE -NOXREF -GEN -NOATR
Print file : QSYSPRT
Library name : -LIBL
Message flagging level : -NONE
Compiler options:
Message limit : -NONE
Replace program object : -YES
User profile : -USER
Authority : -CHANGE
Target Release : -CURRENT
Last change : 10/09/11 08:52:00
Source description : Target C program for Intra
Compiler : IBM ILE C/400 Compiler

Figure D-7 (Part 1 of 3). Target Program Example — CTGTPGM
void main()
{
    // Open the ICF file
    if ((icffptr = _Ropen("CEXAMPLES/TGTICFF", "ar+ indicators=y riofb=y")) == NULL) {
        exit(ERROR);
    }
    // Open the database file
    if ((dbfptr = _Ropen("CEXAMPLES/CUSMSTL", "rr riofb=y")) == NULL) {
        _Rclose(icffptr);
        exit(ERROR);
    }
    // Acquire a session
    _Racquire(icffptr, "ICF/zerodot/zerodot");
    // Check that the acquire was successful and get the record
    if (pos_resp() == NOERROR) {
        // Keep processing numbers until detach or eos is received
        while (process_data() == NOERROR) {
            // Close the ICF file
            _Rclose(icffptr);
        }
    } else {
        // This routine will get a customer number from the source program and
        // attempt to find the corresponding record in the physical file CUSMSTL
        // (by using logical file CUSMSTL). The file is searched using the
        // customer number received as the key. If the record was found, it
        // is sent to the source program, if not a fail is sent.
        _RIOFB_T rio_fdbk; /* Ptr to partial I/O feedback */
        _Rformat(icffptr, "CUST");
        _Rreadn(icffptr, &cust_icf_i, sizeof(cust_icf_i), __DFT);
        // Check if the read was successful, and if so find the
        // record in the data file searching by key
    }
}

int process_data(void);
void send_eos(void);
int pos_resp(void);
if (pos_resp() == NOERROR) {
    /* Read record from database file */
    ria_fdbk = _Rreadk(dbfptr, &cusrec_dbf_i, sizeof(cusrec_dbf_i), 
        __KEY_EQ, cust_icf_i.NUMBER, 5);
    // Check to see if the record was found */
    if (ria_fdbk->num_bytes != 0) {
        memcpy(cinfo_icf_o.CUSTNO, cusrec_dbf_i.PCUST, 5);
        memcpy(cinfo_icf_o.NAME, cusrec_dbf_i.PNAME, 2);
        memcpy(cinfo_icf_o.ADDR, cusrec_dbf_i.PADDR, 2);
        memcpy(cinfo_icf_o.CITY, cusrec_dbf_i.PCITY, 2);
        memcpy(cinfo_icf_o.STATE, cusrec_dbf_i.PSTATE, 2);
        memcpy(cinfo_icf_o.ZIP, cusrec_dbf_i.PZIP, 5);
        memcpy(cinfo_icf_o.ACCBAL, cusrec_dbf_i.PACCBL, 6);
        _Rformat(icffptr, "CINFO");
        _Rwrite(icffptr, &cinfo_icf_o, sizeof(cinfo_icf_o));
    } else {
        // Customer record was not found, send a fail */
        _Rformat(icffptr, "NOCUST");
        _Rwrite(icffptr, NULL, 0);
    }
    // Check for successful return code */
    return(pos_resp());
} else {
    // A detach was received, or a transaction error occurred. */
    return(ERROR);
}

void send_eos() {
    _Rformat(icffptr, "ENDSESSION");
    _Rwrite(icffptr, NULL, 0);
}

/* Check the major return code for a successful operation - 00. */
/* */
pos_resp() {
    comm_fdbk = _Rofbk(icffptr);
    dsp_icf_fdbk = (X100FB_DSP_ICF_T *)((char *)comm_fdbk + 
        comm_fdbk->file_dep_fb_offset);
    if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0)
        return(NOERROR);
    else
        return(ERROR);
}

****** END OF SOURCE ******
Description of the Two-Session Inquiry Program Example

The following explanation describes the transaction between a source program and two target programs, and applies to the COBOL/400 and RPG/400 programs in this appendix.

The source program is started from a display station, and both the display and ICF files are opened. The work station is implicitly acquired when the display file opens, but because the ICF file is created with ACQPGMDEV(*NONE), no ICF devices are acquired during open processing.

The two ICF program devices, ICF00 and ICF01, must be explicitly acquired by the source program using the acquire operation. The source program then starts the two target programs using an evoke function.

The source program uses a specific program device name. Each target program uses an ICF file with a program device name that is associated with the requesting program device. The target program’s only session is the one used to communicate with the source program. When the target program is started, the ICF file is implicitly opened if you are using the RPG/400 language support. However, if you are using the COBOL/400 language support, you need to open the ICF file explicitly using the open operation. Because the file is created with the requesting program device specified on the ACQPGMDEV parameter, the requesting program device is acquired when the ICF file is opened.

The main menu, with a record format CIMENU, is written to the display station and the program waits for a request from the display station. Based on the request made from the display station, the source program uses a write-with-invite function to send an inquiry request to one of the target programs. The target program then sends a reply to the inquiry using a read operation. Finally, the source program sends a detach request and ends the session.

COBOL/400 Source Program for a Two-Session Inquiry

The following describes a COBOL/400 source program for a two-session inquiry.

Program Files: The COBOL/400 two-session source program uses the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTFIL</td>
<td>An ICF file used to send records to and receive records from the target program.</td>
</tr>
<tr>
<td>DSPFIL</td>
<td>A display file used to enter requests that are to be sent to the target program.</td>
</tr>
<tr>
<td>QPRINT</td>
<td>An AS/400 printer file used to print records, both sent and received, as well as major and minor ICF return codes.</td>
</tr>
<tr>
<td>DFILE</td>
<td>An output file that is used to assist in problem analysis for non-recoverable session errors.</td>
</tr>
</tbody>
</table>
**DDS Source:** The DDS for the ICF file (INTFIL) is illustrated in Figure D-8.

Figure D-8. DDS Source for a Two-Session Source Program Using INTFIL
The DDS source file for the display file (DSPFIL) is illustrated in Figure D-9.

Figure D-9 (Part 1 of 2). DDS for Source Program Two-Session Inquiry Using DSPFIL
Figure D-9 (Part 2 of 2). DDS for Source Program Two-Session Inquiry Using DSPFIL

**Configuration:** The following command is needed to create the intrasystem communications device associated with the ICF file:

```
CRTDEVINTR DEVD(INTRADEV)
RMTLOCNAME(INTRARMT)
ONLINE(/C5197NO)
TEXT("THIS IS AN INTRASYSTEM DEVICE DESCRIPTION")
```

**ICF File Creation and Program Device Entry Definition:**

The command needed to create the ICF file is:

```
CRTICFF FILE(INTLIB/INTFIL)
SRCFILE(INTLIB/QINTSRC)
SRCMBR(INTFIL)
ACQPGMDEV(/C5197NONE) MAXPGMDEV(2)
TEXT("SOURCE ICF FILE FOR TWO SESSION PROGRAM")
```

It is not necessary to add a communications entry to the subsystem because the system automatically defines an entry for the device created above when the program is processed. However, the following is an example of what you would use if you decided to add a communications entry:

```
ADDCMNE SBSD(QCMN) DEV(INTRADEV)
```

**Note:** Subsystem QCMN should be stopped before ADDCMNE is entered, and then restarted again. The commands needed to define the two program device entries are:

```
OVRICFDEVE PGMDEV(ICFO0)
RMTLOCNAME(INTRARMT)
FMTSLT(*RECID)
OVRICFDEVE PGMDEV(ICFO1)
RMTLOCNAME(INTRARMT)
FMTSLT(*RECID)
```

The CL program that could be used to run the source program is:

```
CSDINTCL: PGM PARM(&RMT1 &RMT2)
DCL VAR(&RMT1) TYPE(/C5197CHAR) LEN(8)
DCL VAR(&RMT2) TYPE(/C5197CHAR) LEN(8)
CHGJOB OUTQ(INTLIB/INTOUTQ)
LOG(4 /C5197SECLVL)
LOGCLPGM(*YES)
OVRICFDEVE PGMDEV(ICFO0)
RMTLOCNAME(&RMT1)
FMTSLT(*RECID)
OVRICFDEVE PGMDEV(ICFO1)
RMTLOCNAME(&RMT2)
FMTSLT(*RECID)
CALL INTLIB/CSDINT
ENDCSDINTCL: ENDPGM
```
**Program Explanation:** The following explains the structure of the program example illustrated in Figure D-10 on page D-20. The ICF file used in the example is defined by the user, and uses externally described data formats (DDS). The reference numbers in the explanation below correspond to the numbers in the following program example.

All output operations to the ICF file in the example are done using the write statement with the record format name coded as an operand.

1. This section defines the ICF file (INTFIL) and the display file (DSPFIL) used in the program. INTFIL is the ICF file used to send records to and receive records from each of the two target programs. INTFIL is established using the file-level keyword, INDARA, indicating that a separate indicator area is used. DSPFIL is the display file used to receive user’s requests and to report the information received based on the request.

   The control area clause in the select statements of INTFIL and DSPFIL is used to define the I/O feedback area. Information from the I/O feedback is used to determine the major/minor return code, record format, and function key pressed.

2. THE DSP-ERROR SECTION and CMN-ERROR SECTION define the error handling procedures for I/O errors on the DSPFIL and INTFIL. A DSPFIL I/O error causes the program to end, and an error message to be sent to the printer file. The section for INTFIL file I/O errors checks the major/minor return code to determine if the error is recoverable. If the error is recoverable (major code 83), it sets a flag (ERR-SW) to 1 and returns to the program.

3. The program opens the files to be used and initializes the ICF file separate indicator area.

4. If the ERR-SW switch is set to 1, indicating that a recoverable error has occurred, the program determines whether the open-retry count limit of nine has been exceeded. If it has, the program goes to section 19 and then ends. If the limit count is less than nine, one is added to the count and control passes to section 17 and then to section 3 to try to open the file.

5. The two program devices used by the program are explicitly acquired.

6. A read operation is issued to the display device, and the program waits for an input request from the user. When a record is returned, the last record format used (as specified in the RCD-FMT field in the I/O control area) is checked. Based on the value in RCD-FMT, the program branches to the appropriate routine.

   If a match is not found for the display record format, the main menu (CIMENU) is written to the work station and control is returned to section 6.

7. The MENU routine is called if the request is made from the main menu (CIMENU). If the CMD-KEY variable is set to 01, indicating that the operator pressed function key 1, the two transactions and sessions are ended and the program ends. If the operator entered option 1, the program writes the Item Inquiry menu (ITMNU) to the work station and returns to section 6.

   If the option is not 1, the Buyer Inquiry menu (DTLMNU) is written to the work station and control is passed to section 6.

8. The ITMIN routine is called when the user is requesting an item inquiry (record format ITMNU). If function key 1 (CMD-KEY = 01) is pressed, control passes to section 19, and then to section 20, the two transactions end, and the program ends. If function key 2 is pressed, the inquiry request is canceled, the main menu (CMENU) is written to the work station, and the program returns to section 16.

   The item number is read from the work station and then the request is sent to the target program on program device ICF01.

   The request is sent to the appropriate target program by writing data to the program device using format ITMREQ. The INVITE keyword is specified as part of the ITMREQ format to give the target program permission to send.

   A timer is issued for 30 seconds before the read operation. This is provided to allow the local program to have a time out when no response is received from the target program.

   The read is an implied read-from-invited-program devices because no record format is specified in the read statement.
If a fail indication is received (the item number requested was not found), the request is not validated and a new item inquiry menu (ITMMNU) is written to the display device.

Control goes to section 9 to process the item information based on the input data that was received, and the result is written to the display using format ITMSC2.

After returning from section 9, the program returns to section 6.

The routine ITMOUT is called when the target program responds to a request for an item record. If the returned response is a fail indication (checked in section 8), the request is invalidated and a new Item Inquiry menu (ITMMNU) is written to the work station.

The program then performs the calculations to set the quantity fields and writes the result to the requesting work station using record format ITMSC2.

The program then returns to the calling routine.

The routine ITMRTN is called to process the next user request. If function key 1 (CMD-KEY = 01) is pressed, the transactions and session are ended in section 19, and control goes to section 20 to end the program.

If function key 2 is pressed, the main menu (CMENU) is written to the work station. If function key 3 is pressed, the Item Inquiry menu is written to the work station, and the program returns to section 6. By pressing Enter, the profit and loss figures are calculated and written to the work station before returning control to section 6.

The PROFIT-LOSS routine calculates the profit and loss figures for the second display of the requested item number.

The DTLIN routine is called when a request is read from the Buyer Inquiry menu (DTLMNU). If function key 1 (CMD-KEY = 01) is pressed, the transactions and sessions are ended. If function key 2 (CMD-KEY = 02) is pressed, the main menu (CMENU) is written to the work station and the program returns to section 6.

The buyer inquiry request is sent to the target program by writing data to the program device ICF00 using format DTLREQ. The INVITE keyword is specified as part of the DLTREQ format to give the target program permission to send.

Control goes to section 14 to retrieve the buyer detail information.

Routine DTLRTN in section 14 is called to continue the buyer information processing.

The program then returns to section 6.

The routine DTLRTN is called from section 6, and handles the user’s request following the display of the buyer information. Function key 1 ends the job, function key 2 displays the main menu (CMENU), and pressing Enter displays the Buyer Inquiry menu (DTLMNU). Control then returns to section 6.

The CUSTOMER-DETAIL routine issues the read operation to the program device.

This read is an implied read-from-invited-program-devices because no record format is specified on the read statement.

A check is made of the MAJ-MIN return code for possible error conditions on a successful return (control is automatically passed to section 2 for unsuccessful I/O operations). A 0310 return code means the remote program has timed out. (The timer was issued on the write operation.) If no data was received (return codes of 03xx), the request is sent again to the remote program. Finally, if the data returns in the wrong format, control is passed to section 17.

The buyer information received from the target program is processed, and the result is written to the user work station using screen format DTLBLK.

Control returns to the calling routine.

The EVOKE routine builds the evoke requests to send to the remote programs. Because the DDS keyword for the record format only specifies the field identifiers with the record, this code moves the literal value CTDINTCL to the field PGMID, and INTLIB to the field LIB.

When the program start request is received at the remote program, INTLIB is searched for CTDINTCL and that program then starts. CTDINTCL is a CL program that contains CL statements, as illustrated on D-17.

The ERROR-RECOVERY routine ends the transactions and closes the files. The ERR-SW indicator is set again, and control returns to the calling routine.

The EXIT-FORMAT-ERR routine is run when the program detects data in an incorrect record format. It writes an error message to the printer file, ends the program, and implicitly ends the session.

The DETACH routine issues the detach function to the ICF file for each of the two program devices. In the program using the user-supplied format, the write operation is issued using the record format name DETACH.

The END-JOB routine releases the program devices and close the files. The program ends.
**Figure D-10 (Part 1 of 13). Source Program Example — CSDINT**
Appendix D. Program Examples

Figure D-10 (Part 2 of 13). Source Program Example — CSDINT

```
88 +/zerodot/zerodot/zerodot/zerodot71 /zerodot6 FILL2 PIC X(57). <-ALL-FMTS
87 +/zerodot/zerodot/zerodot/zerodot7/zerodot /zerodot6 IDEPT PIC S9(3). <-ALL-FMTS
86 +/zerodot/zerodot/zerodot/zerodot69 /zerodot6 DSTTYD PIC S9(11). <-ALL-FMTS
85 +/zerodot/zerodot/zerodot/zerodot68 /zerodot6 DSPM/zerodot3 PIC S9(9). <-ALL-FMTS
84 +/zerodot/zerodot/zerodot/zerodot67 /zerodot6 DSPM/zerodot2 PIC S9(9). <-ALL-FMTS
83 +/zerodot/zerodot/zerodot/zerodot66 /zerodot6 DSPM/zerodot1 PIC S9(9). <-ALL-FMTS
82 +/zerodot/zerodot/zerodot/zerodot65 /zerodot6 DSLSTM PIC S9(9). <-ALL-FMTS
79 +/zerodot/zerodot/zerodot/zerodot62 /zerodot6 CUSTNO PIC S9(6). <-ALL-FMTS
78 +/zerodot/zerodot/zerodot/zerodot61 /zerodot6 RECCUS PIC X(1). <-ALL-FMTS
77 +/zerodot/zerodot/zerodot/zerodot6/zerodot /zerodot5 DTLRSP-O REDEFINES INTFIL-RECORD. <-ALL-FMTS
76 +/zerodot/zerodot/zerodot/zerodot58/c5197 OUTPUT FORMAT:DTLRSP FROM FILE INTFIL OF LIBRARY INTLIB <-ALL-FMTS
75 +/zerodot/zerodot/zerodot/zerodot57/zerodot /zerodot6 DSLSTY PIC S9(9)V9(2). <-ALL-FMTS
74 +/zerodot/zerodot/zerodot/zerodot56/c5197 OUTPUT FORMAT:ITMRSP FROM FILE INTFIL OF LIBRARY INTLIB <-ALL-FMTS
73 +/zerodot/zerodot/zerodot/zerodot5/zerodot /zerodot6 SLSTY PIC S9(9)V9(2). <-ALL-FMTS
72 +/zerodot/zerodot/zerodot/zerodot49/zerodot /zerodot6 DNAME PIC X(3/zerodot). <-ALL-FMTS
71 +/zerodot/zerodot/zerodot/zerodot48/zerodot /zerodot6 CUSTNO PIC S9(6). <-ALL-FMTS
70 +/zerodot/zerodot/zerodot/zerodot47/zerodot /zerodot6 RECCUS PIC X(1). <-ALL-FMTS
69 +/zerodot/zerodot/zerodot/zerodot46/zerodot /zerodot5 DTLRSP-I REDEFINES INTFIL-RECORD. <-ALL-FMTS
68 +/zerodot/zerodot/zerodot/zerodot44/c5197 INPUT FORMAT:DTLRSP FROM FILE INTFIL OF LIBRARY INTLIB <-ALL-FMTS
67 +/zerodot/zerodot/zerodot/zerodot43/zerodot /zerodot6 FILL2 PIC X(56). <-ALL-FMTS
66 +/zerodot/zerodot/zerodot/zerodot42/zerodot /zerodot6 LOS PIC S9(7)V9(2). <-ALL-FMTS
65 +/zerodot/zerodot/zerodot/zerodot41/zerodot /zerodot6 PRO PIC S9(3)V9(2). <-ALL-FMTS
64 +/zerodot/zerodot/zerodot/zerodot40/zerodot /zerodot6 CSTTY PIC S9(9)V9(2). <-ALL-FMTS
63 +/zerodot/zerodot/zerodot/zerodot39/zerodot /zerodot6 CSTTM PIC S9(7)V9(2). <-ALL-FMTS
62 +/zerodot/zerodot/zerodot/zerodot38/zerodot /zerodot6 SLSTY PIC S9(9)V9(2). <-ALL-FMTS
61 +/zerodot/zerodot/zerodot/zerodot37/zerodot /zerodot6 SLSTM PIC S9(7)V9(2). <-ALL-FMTS
60 +/zerodot/zerodot/zerodot/zerodot36/zerodot /zerodot6 UFRT PIC S9(3)V9(2). <-ALL-FMTS
59 +/zerodot/zerodot/zerodot/zerodot35/zerodot /zerodot6 PR/zerodot5 PIC S9(7). <-ALL-FMTS
58 +/zerodot/zerodot/zerodot/zerodot34/zerodot /zerodot6 PR/zerodot1 PIC S9(5)V9(2). <-ALL-FMTS
57 +/zerodot/zerodot/zerodot/zerodot33/zerodot /zerodot6 UNITQ PIC X(2). <-ALL-FMTS
56 +/zerodot/zerodot/zerodot/zerodot32/zerodot /zerodot6 QTYBO PIC S9(7). <-ALL-FMTS
55 +/zerodot/zerodot/zerodot/zerodot31/zerodot /zerodot6 QTYOO PIC S9(7). <-ALL-FMTS
54 +/zerodot/zerodot/zerodot/zerodot30/zerodot /zerodot6 QTYOH PIC S9(7). <-ALL-FMTS
52 +/zerodot/zerodot/zerodot/zerodot28/zerodot /zerodot6 ITEMNO PIC S9(6). <-ALL-FMTS
51 +/zerodot/zerodot/zerodot/zerodot27/zerodot /zerodot6 RECITM PIC X(1). <-ALL-FMTS
50 +/zerodot/zerodot/zerodot/zerodot26/zerodot /zerodot6 DESC PIC X(30). <-ALL-FMTS
49 +/zerodot/zerodot/zerodot/zerodot25/zerodot /zerodot6 QTYOH PIC S9(7). <-ALL-FMTS
48 +/zerodot/zerodot/zerodot/zerodot24/zerodot /zerodot6 QTYDD PIC S9(6). <-ALL-FMTS
47 +/zerodot/zerodot/zerodot/zerodot23/zerodot /zerodot6 QTYMM PIC S9(6). <-ALL-FMTS
46 +/zerodot/zerodot/zerodot/zerodot22/zerodot /zerodot6 FILL1 PIC X(56). <-ALL-FMTS
45 +/zerodot/zerodot/zerodot/zerodot21/zerodot /zerodot6 LOS PIC S9(7)V9(2). <-ALL-FMTS
44 +/zerodot/zerodot/zerodot/zerodot20/zerodot /zerodot6 PRO PIC S9(3)V9(2). <-ALL-FMTS
43 +/zerodot/zerodot/zerodot/zerodot19/zerodot /zerodot6 CSTTY PIC S9(9)V9(2). <-ALL-FMTS
42 +/zerodot/zerodot/zerodot/zerodot18/zerodot /zerodot6 CSTTM PIC S9(7)V9(2). <-ALL-FMTS
41 +/zerodot/zerodot/zerodot/zerodot17/zerodot /zerodot6 SLSTY PIC S9(9)V9(2). <-ALL-FMTS
40 +/zerodot/zerodot/zerodot/zerodot16/zerodot /zerodot6 SLSTM PIC S9(7)V9(2). <-ALL-FMTS
39 +/zerodot/zerodot/zerodot/zerodot15/zerodot /zerodot6 UFRT PIC S9(3)V9(2). <-ALL-FMTS
38 +/zerodot/zerodot/zerodot/zerodot14/zerodot /zerodot6 PR/zerodot5 PIC S9(7). <-ALL-FMTS
37 +/zerodot/zerodot/zerodot/zerodot13/zerodot /zerodot6 PR/zerodot1 PIC S9(5)V9(2). <-ALL-FMTS
36 +/zerodot/zerodot/zerodot/zerodot12/zerodot /zerodot6 UNITQ PIC X(2). <-ALL-FMTS
35 +/zerodot/zerodot/zerodot/zerodot11/zerodot /zerodot6 QTYBO PIC S9(7). <-ALL-FMTS
34 +/zerodot/zerodot/zerodot/zerodot10/zerodot /zerodot6 QTYOO PIC S9(7). <-ALL-FMTS
33 +/zerodot/zerodot/zerodot/zerodot9/zerodot /zerodot6 QTYOH PIC S9(7). <-ALL-FMTS
32 +/zerodot/zerodot/zerodot/zerodot8/zerodot /zerodot6 DESC PIC X(30). <-ALL-FMTS
31 +/zerodot/zerodot/zerodot/zerodot7/zerodot /zerodot6 QTYMM PIC S9(6). <-ALL-FMTS
30 +/zerodot/zerodot/zerodot/zerodot6/zerodot /zerodot6 QTYDD PIC S9(6). <-ALL-FMTS
29 +/zerodot/zerodot/zerodot/zerodot5/zerodot /zerodot6 QTYMM PIC S9(6). <-ALL-FMTS
28 +/zerodot/zerodot/zerodot/zerodot4/zerodot /zerodot6 QTYOH PIC S9(7). <-ALL-FMTS
26 +/zerodot/zerodot/zerodot/zerodot2/zerodot /zerodot6 QTYMM PIC S9(6). <-ALL-FMTS
25 +/zerodot/zerodot/zerodot/zerodot1/zerodot /zerodot6 QTYOH PIC S9(7). <-ALL-FMTS
```

Figure D-10 (Part 2 of 13). Source Program Example — CSDINT
**Figure D-10 (Part 3 of 13). Source Program Example — CSDINT**
Figure D-10 (Part 4 of 13). Source Program Example — CSDINT
Intrasystem Communications Programming V4R1

Figure D-10 (Part 5 of 13). Source Program Example — CSDINT

+000008+  OUTPUT FORMAT:ITMSC3 FROM FILE DSPFIL OF LIBRARY INTLIB <-ALL-FMTS
+000081+  ITEM INQUIRY SCREEN 3 <-ALL-FMTS
158 +000082  05 ITMSC3-O REDEFINES DSPFIL-RECORD. <-ALL-FMTS
159 +000083  06 SLSM PIC 9(7)V9(2). <-ALL-FMTS
160 +000084  06 SLSY PIC 9(7)V9(2). <-ALL-FMTS
161 +000085  06 CSTM PIC 9(7)V9(2). <-ALL-FMTS
162 +000086  06 CSTY PIC 9(7)V9(2). <-ALL-FMTS
163 +000087  06 PROFIT PIC 9(7)V9(2). <-ALL-FMTS
164 +000088  06 LOSTS PIC 9(7)V9(2). <-ALL-FMTS
+000089+  INPUT FORMAT:TIMOUT FROM FILE DSPFIL OF LIBRARY INTLIB <-ALL-FMTS
+000090+  TIME OUT SCREEN <-ALL-FMTS
165 +000091  05 TIMOUT-I REDEFINES DSPFIL-RECORD. <-ALL-FMTS
166 +000092  06 TIMOUT-I-INDIC. <-ALL-FMTS
167 +000093  07 IN98 PIC 1 INDIC 98. <-ALL-FMTS
168 +000094  07 IN99 PIC 1 INDIC 99. <-ALL-FMTS
169 +000095  07 IN97 PIC 1 INDIC 97. <-ALL-FMTS
170 +000096  06 TIMRSP PIC X(1). <-ALL-FMTS
+000097+  OUTPUT FORMAT:TIMOUT FROM FILE DSPFIL OF LIBRARY INTLIB <-ALL-FMTS
+000098+  TIME OUT SCREEN <-ALL-FMTS
+000099+  05 TIMOUT-D REDEFINES DSPFIL-RECORD. <-ALL-FMTS
171 005200  01 PRINTREC. 09/30/87
172 005300  01 PRINTREC. 09/30/87
173 005400  01 PRINTREC. 01/14/88
174 005500  05 RC PIC 9999. 01/15/88
175 005600  05 ERRMSG PIC X(128). 01/14/88
176 005700  WORKING-STORAGE SECTION. 09/30/87
177 005800  77 STATUS-IND PIC X(2). 09/30/87
178 005900  77 STATUS-ISP PIC X(2). 09/30/87
179 006000  77 MAJ-MIN-SAY PIC X(4). 09/30/87
180 006100  77 EOF-FILE-SW PIC X VALUE "0". 09/30/87
181 006200  77 ERR-FILE-SW PIC X VALUE "0". 09/30/87
182 006300  77 INDOFF PIC 1 INDIC 97. 11/16/88
183 006400  INPUT "CPYF HEXDUMP +LIST PRTFMT(+HEX)". 09/30/87
184 006500  77 OPEN-COUNT PIC 9(1) VALUE 0. 09/30/87
185 006600  77 LEN PIC 9(10)V9(5) COMP. 09/30/87
186 006700  77 PROFMT PIC 9(7)V9(2) COMP-4. 09/30/87
187 006800  77 CMDAREA PIC X(31). 09/30/87
188 006900  VALUE "CPYF HEXDUMP +LIST PRTFMT(+HEX)". 09/30/87
189 007000  01 SUBKEY-VALUE. 09/30/87
190 007100  05 SUBKEY PIC 9(3) VALUE 0. 09/30/87
191 007200  01 TR-CTL-AREA. 09/30/87
192 007300  05 FILLER PIC X(2). 09/30/87
193 007400  05 PGM-DEV-NME PIC X(10). 09/30/87
194 007500  05 RCD-FMT-NME PIC X(10). 09/30/87
195 007600  01 ENTF-INDIC-AREA. 11/21/88
196 007700  05 IN95. PIC 1 INDIC 25. 11/16/88
197 007800  88 IN95-ON VALUE B"0". 11/16/88
198 007900  88 IN95-OFF VALUE B"1". 11/16/88
199 008000  05 IN90 PIC 1 INDIC 90. 11/16/88
200 008100  88 IN90-ON VALUE B"0". 11/16/88
201 008200  88 IN90-OFF VALUE B"1". 11/16/88
202 008300  01 DSPF-INDIC-AREA. 09/30/87
203 008400  05 IN23 PIC 1 INDIC 23. 09/30/87
204 008500  88 IN23-ON VALUE B"1". 09/30/87
205 008600  88 IN23-OFF VALUE B"0". 09/30/87
206 008700  05 IN97 PIC 1 INDIC 97. 09/30/87
207 008800  88 IN97-ON VALUE B"1". 09/30/87
208 008900  88 IN97-OFF VALUE B"0". 09/30/87
209 009000  05 IN98 PIC 1 INDIC 98. 09/30/87
210 009100  88 IN98-ON VALUE B"1". 09/30/87
211 009200  88 IN98-OFF VALUE B"0". 09/30/87
212 009300  05 IN99 PIC 1 INDIC 99. 09/30/87
213 009400  88 IN99-ON VALUE B"1". 09/30/87
214 009500  88 IN99-OFF VALUE B"0". 09/30/87
215 009600  01 MAJ-MIN. 09/30/87
216 009700  05 MAJ PIC X(2). 09/30/87
217 009800  05 MIN PIC X(2). 09/30/87
218 009900  01 DISPLAY-FEEDBACK. 09/30/87
219 010000  05 CMD-KEY PIC X(2). 09/30/87
220 010100  05 FILLER PIC X(10). 09/30/87
221 010200  05 RCD-FMT PIC X(10). 09/30/87
+010300+  01 PROCEDURE DIVISION. 11/16/88
+010400  01 PROCEDURE DIVISION. 09/30/87

Figure D-10 (Part 5 of 13). Source Program Example — CSDINT
Figure D-10 (Part 6 of 13). Source Program Example — CSDINT
Figure D-10 (Part 7 of 13). Source Program Example — CSDINT

Figure D-10 (Part 7 of 13). Source Program Example — CSDINT
Figure D-10 (Part 8 of 13). Source Program Example — CSDINT

```
023200+ 4
023300 READQ.

259 023400 READ DSPFIL INDICATORS ARE DSPF-INDIC-AREA.
09/30/87
260 023500 IF RCD-FMT = "CIMENU"
09/30/87
261 023600 PERFORM MENU-Routine THRU MENU-EXIT
09/30/87
262 023700 GO TO READQ.
09/30/87
263 023800 IF RCD-FMT = "ITMMNU"
09/30/87
264 023900 PERFORM ITMIN-Routine THRU ITMIN-EXIT
09/30/87
265 024000 GO TO READQ.
09/30/87
266 024100 IF RCD-FMT = "ITMSC3"
09/30/87
267 024200 PERFORM ITMRRT-ROUTINE THRU ITMRRT-EXIT
09/30/87
268 024300 GO TO READQ.
09/30/87
269 024400 IF RCD-FMT = "ITMSC3"
09/30/87
270 024500 PERFORM ITMRRT-ROUTINE THRU ITMRRT-EXIT
09/30/87
271 024600 GO TO READQ.
09/30/87
272 024700 IF RCD-FMT = "DTLMNU"
09/30/87
273 024800 PERFORM DTLIN-Routine THRU DTLIN-EXIT
09/30/87
274 024900 GO TO READQ.
09/30/87
275 025000 IF RCD-FMT = "DTLMNU"
10/12/87
276 025100 PERFORM DTLRTN-ROUTINE THRU DTLRTN-EXIT
10/12/87
277 025200 GO TO READQ.
10/12/87
278 025300 WRITE DSPREC FORMAT IS "CIMENU".
09/30/87
279 025400* 11/18/88
280 025500 GO TO READQ.
09/30/87
281 025600* 11/18/88
025700******************************************************************************
09/30/87
025800* * 09/30/87
025900* * 09/30/87
026000* * 09/30/87
026100* THE MAIN MENU IS READ TO DETERMINE THE REQUEST ENTERED
10/12/87
026200* BY THE USER. IF CMD 1 (=999) IS PRESSED, THE PROGRAM IS
10/12/87
026300* ENDED. IF OPTION = 1, AN ITEM INQUIRY MENU IS WRITTEN TO
10/12/87
026400* TO SCREEN, IF OPTION = 2, A CUSTOMER INQUIRY MENU IS
10/12/87
026500* WRITTEN TO THE SCREEN.
* 10/12/87
026600* 09/30/87
026700******************************************************************************
09/30/87
026800+ 7
026900 MENU-Routine.
09/30/87
280 027000 IF CMD-KEY = "Q1"
09/30/87
281 027100 PERFORM DETACH-ROUTINE THRU DETACH-EXIT
09/30/87
282 027200 GO TO END-JOB.
09/30/87
283 027300 IF OPTION = "O"
09/30/87
284 027400 WRITE DSPREC FORMAT IS "ITMMNU"
09/30/87
285 027500 ELSE
09/30/87
286 027600 WRITE DSPREC FORMAT IS "DTLMNU".
09/30/87
027700 MENU-EXIT.
09/30/87
027800 EXIT.
09/30/87
027900* 11/18/88
028000******************************************************************************
09/30/87
028100* * 09/30/87
028200* 09/30/87
028300* 09/30/87
028400******************************************************************************
09/30/87
573BC1B VR1MO 91052A 8A/400 COBOL/400 Source INTLIB/CSINT RCHB321 10/08/90 11:09:19 Page 11
STMT SEQNR -A 1 B-2 C-3 D-4 E-5 F-6 G-7. IDENTFCN S COPYNAME CHG DATE
026400+ THE ITEM NUMBER REQUESTED BY THE USER ON THE ITEM INQUIRY
09/30/87
026500+ SCREEN IS CHECKED. THIS IS DETERMINED BY THE
09/30/87
026600+ DISPLAY RECORD FORMAT BEING PROCESSED - IN THIS CASE ITMMNU.
09/30/87
026700+ 03/16/89
026800+ IF CMD KEY 1 IS PRESSED, THE PROGRAM IS ENDED. IF CMD KEY 2
10/11/87
026900+ IS PRESSED, THE ITEM INQUIRY REQUEST IS CANCELED, AND THE
09/30/87
029000+ MAIN MENU (CIMENU) IS WRITTEN TO THE SCREEN.
* 09/30/87
029100+ 09/30/87
029200+ IF AN ITEM NUMBER IS ENTERED, A ITEM INQUIRY REQUEST IS
09/30/87
029300+ SENT TO THE APPROPRIATE TARGET PROGRAM.
* 10/05/90
029400+ 09/30/87
029500+ A CHECK IS MADE FOR THREE CONDITIONS FOLLOWING THE READ.
* 10/14/87
029600+ 1) THE TARGET PROGRAM TIMED OUT, 2) NO DATA RECEIVED, AND
10/05/90
029700+ 3) DATA RETURNED IN AN UNEXPECTED RECORD FORMAT.
* 10/14/87
029800+ 10/14/87
029900+ IF THE TIMER RUNS OUT (MAJ-MIN = 0310) A MESSAGE
11/21/88
030000+ IS WRITTEN TO THE SCREEN, ASKING TO TRY AGAIN OR END THE
* 10/14/87
030100+ PROGRAM.
10/14/87
030200+ 11/21/88
```
Figure D-10 (Part 9 of 13). Source Program Example — CSDINT

030000+ IF A RECEIVE FAIL INDICATION IS RECEIVED (IN-25 FLAG ON),  * 11/21/88
0304000 AFTER THE READ OPERATION TO THE PROGRAM DEVICE,  * 11/21/88
030500+ A FRESH ITEM MENU (ITMMNU) IS WRITTEN  * 11/21/88
030600+ TO THE DISPLAY DEVICE.  * 11/21/88
0307000+ IF NO DATA IS RECEIVED OR IF RECEIVE FAIL INDICATION  * 10/14/87
0309000+ IS RECEIVED (IN-25 FLAG IS ON), AFTER THE READ OPERATION  * 11/21/88
031100+ TO THE PROGRAM DEVICE, THE REQUEST IS SENT AGAIN  * 10/05/90
0311000+ AND THE READ OPERATION IS ISSUED TO THE PROGRAM DEVICE.  * 10/05/90
0314000+ IF THE RECORD RETURNS WITH THE WRONG RECORD FORMAT, THE  * 10/14/87
0313000+ PROGRAM WILL GO TO EXIT-FORMAT-ERR ROUTINE.  * 10/14/87
0315000*  * 10/14/87
031600+******************************************************************************* 09/30/87
0317000 3 09/30/87
286 031800 ITMIN-Routine. 09/30/87
287 031900 IF CMD-KEY = "01" 09/30/87
288 032000 PERFORM DETACH-Routine THRU DETACH-EXIT 10/12/87
289 032100 GO TO END-JOB. 10/12/87
290 032200 IF CMD-KEY = "02" 10/12/87
291 032300 WRITE DSPREC FORMAT IS "CIMENU" 10/12/87
292 032400 GO TO ITMIN-EXIT. 10/12/87
293 032500 ITMIN. 11/18/88
293 032600 MOVE CORR ITMMNU-I TO ITMREQ-O. 11/18/88
* ** CORRESPONDING items for statement 293:
* ** ITEMNO
* ** End of CORRESPONDING items for statement 293
294 032700 MOVE "ICF01 " TO PGM-DEV-NME. 09/30/87
295 032800 MOVE ZEROS TO INTF-INDIC-AREA. 11/21/88
296 032900 WRITE INTREC FORMAT IS "ITMREQ" 11/21/88
033000 TERMINAL IS PGM-DEV-NME. 09/30/87
033100 TRY-AGAIN. 10/01/87
297 033200 MOVE "ICF01 " TO PGM-DEV-NME. 10/09/89
298 033300 MOVE ZEROS TO INT-INDIC-AREA. 11/28/88
299 033400 WRITE INTREC FORMAT IS "ITMRSP" 11/28/88
033500 TERMINAL IS PGM-DEV-NME. 11/28/88
5738CB1 V2R1M 910524 AS/400 COBOL/400 Source INTLIB/CSDINT RCH0321 10/08/90 11:09:19 Page 12
STM SEQNR A 1 B..+....2....+....3....+....4....+....5....+....6....+....7....IDENTFCN S COPYNAME CHG DATE
300 033600 READ INTFIL 10/09/90
033700 INDICATORS ARE INTF-INDIC-AREA. 10/09/90
301 033800 IF MAJ-MIN = "0310" 10/01/87
302 033900 WRITE DSPREC FORMAT IS "TIMOUT" 09/30/87
303 034000 READ DSPFIL INDICATORS ARE DSPF-INDIC-AREA 10/12/87
304 034100 IF TIMRSP = "1" GO TO TRY-AGAIN END-IF 01/21/88
306 034200 IF TIMRSP = "2" GO TO END-JOB END-IF. 01/21/88
308 034300 IF IN25-ON 11/16/88
309 034400 WRITE DSPREC FORMAT IS "ITMMNU" 11/16/88
310 034500 GO TO ITMIN-EXIT. 11/16/88
311 034600 IF RCD-FMT-NME IS NOT EQUAL "ITMRSP" GO TO EXIT-FORMAT-ERR. 11/28/88
313 034700 PERFORM ITMOUT-Routine THRU ITMOUT-EXIT. 09/30/87
034800 ITMIN-EXIT. 09/30/87
034900 EXIT. 09/30/87
035000+ 11/18/88
035100+******************************************************************************* 09/30/87
035200+ * 09/30/87
035300+ PROCESS ITEM INFORMATION 09/30/87
035400+ * 09/30/87
035500+ THE ITEM RECORD RECEIVED FROM THE TARGET PROGRAM AND THE 09/30/87
035600+ INFORMATION ABOUT THE ITEM IS PROCESSED AND DISPLAYED. 09/30/87
035700+ IF ITEMNO IS 0 OR LESS, IT IS AN INVALID REQUEST AND A FRESH 09/30/87
035800+ ITEM MENU IS WRITTEN TO THE SCREEN. IF THE REQUEST IS 09/30/87
035900+ VALID, VALUES ARE CALCULATED BASED ON THE INFORMATION 09/30/87
036000+ RECEIVED. 09/30/87
036100+ * 09/30/87
036200+******************************************************************************* 09/30/87

Figure D-10 (Part 9 of 13). Source Program Example — CSDINT
Figure D-10 (Part 10 of 13). Source Program Example — CSDINT
Figure D-10 (Part 1 of 13). Source Program Example — CSDINT
Appendix D. Program Examples

Figure D-10 (Part 12 of 13). Source Program Example — CSDINT

053600+ 03/17/89
053700+ 10/08/90
370 053800 CUSTOMER-DETAIL.
371 053900 MOVE ZEROS TO INTF-INDIC-AREA.
372 054000 MOVE "ICFO" TO PGM-DEV-NME.
373 054100 WRITE INTRC FORMAT IS "TIMERR".
374 054200 TERMINAL IS PGM-DEV-NME.
375 054300 READ INTR.
376 054400 INDICATORS ARE INTF-INDIC-AREA.
377 054500 IF MAJ-MIN = "0310" 10/01/87
378 054600 WRITE DSPREC FORMAT IS "TIMIN".
379 054700 READ DSPFIL INDICATORS ARE DSPF-INDIC-AREA.
380 054800 IF TIMRSP = "1" GO TO CUSTOMER-DETAIL END-IF.
381 054900 IF TIMRSP = "2" GO TO END-JOB END-IF.
382 055000 IF IN25-ON
383 055100 WRITE DSPREC FORMAT IS "CIMENU".
384 055200 GO TO CUSTOMER-DETAIL-EXIT.
385 055300 IF MAJ = "03" 09/16/89
386 055400 MOVE ZEROS TO INTF-INDIC-AREA.
387 055500 WRITE INTRC FORMAT IS "DILREQ".
388 055600 TERMINAL IS PGM-DEV-NME.
389 055700 GO TO CUSTOMER-DETAIL.
390 055800 MOVE CUSTNO OF DTLRSP-I TO CUSTNM OF DTLSCR-Q.
391 055900 MOVE DNAME OF DTLRSP-I TO CNAME OF DTLSCR-Q.
392 056000 MOVE DLSRN OF DTLRSP-I TO DLSRN OF DTLSCR-Q.
393 056100 MOVE DSLSRN OF DTLRSP-I TO DSLSRN OF DTLSCR-Q.
394 056200 MOVE DSPM1 OF DTLRSP-I TO DSPM1 OF DTLSCR-Q.
395 056300 MOVE DSPM2 OF DTLRSP-I TO DSPM2 OF DTLSCR-Q.
396 056400 MOVE DSTDYD OF DTLRSP-I TO DSTDYD OF DTLSCR-Q.
397 056500 MOVE DIPDT OF DTLRSP-I TO DEPT OF DTLSCR-Q.
398 056600 WRITE DSPREC FORMAT IS "DILSCR".
399 056700 CUSTOMER-DETAIL-EXIT.
400 056800 EXIT.
401 056900 IF TIMRSP = "1" GO TO CUSTOMER-DETAIL END-IF.
402 057000 IF MAJ-MIN = "01/18/88"
403 057100 THE EVOKE-Routine IS CALLED TO EVOKE THE TARGET PROGRAM.
404 057200 THE SAME TARGET PROGRAM (INTLIB/CTDINTCL) IS EVOKEC TWICE.
405 057300 CREATE TWO DIFFERENT JOBS. THE PROGRAM DEVICE IS USED TO.
406 057400 IDENTIFY THEM.
407 057500 IDENTIFY THEM.
408 057600 IF MAJ = "03" 09/30/87
409 057700 WRITE INTRC FORMAT IS "DILREQ".
410 057800 TERMINAL IS PGM-DEV-NME.
411 057900 CUSTOMER-DETAIL.
412 058000 EVOKE-Routine.
413 058100 MOVE "CTDINTCL" TO PGMID OF EVKREQ-Q.
414 058200 MOVE "INTLIB" TO LIB OF EVKREQ-Q.
415 058300 MOVE "ICFO" TO PGM-DEV-NME.
416 058400 WRITE INTRC FORMAT IS "EVKREQ".
417 058500 TERMINAL IS PGM-DEV-NME.
418 058600 MOVE "ICFO" TO PGM-DEV-NME.
419 058700 WRITE INTRC FORMAT IS "EVKREQ".
420 058800 TERMINAL IS PGM-DEV-NME.
421 058900 EVOKE-EXIT.
422 059000 EXIT.
423 059100+ 10/08/90
424 059200+ 03/16/89
425 059300+ 03/16/89
426 059400+ 03/16/89
427 059500+ 03/16/89
428 059600+ 03/16/89
429 059700+ 03/16/89
430 059800+ 10/08/90
431 059900+ 09/30/87
432 060000+ 09/30/87
433 060100 CLOSE INTFIL DSPFIL.
434 060200 OPRINT.
435 060300 MOVE "0" TO ERR-SW.
436 060400 ERROR-RECOVERY-EXIT.
437 060500 EXIT.
438 060600+ 03/16/89
439 060700+ 03/16/89
440 060800+ UNEXPECTED REG-FMT-NME IN THE I-O-FEEDBACK AREA FOR INTFIL.
441 060900+ AN ERROR MESSAGE IS PRINTED AND THE PROGRAM ENDS.
442 061000+ AN ERROR MESSAGE IS PRINTED AND THE PROGRAM ENDS.
443 061100+ 03/16/89
444 061200+ 03/16/89

Figure D-10 (Part 12 of 13). Source Program Example — CSDINT
061300+ 32 10/08/90
409 061400 EXIT-FORMAT-ERR. 10/01/87
410 061500 MOVE MAJ-MIN TO RC. 01/14/88
411 061600 MOVE "RECORD FORMAT IS INCORRECT ON READ" * 10/01/87
061700 TO ERRMSG. 01/14/88
412 061800 WRITE PRINTREC. 10/01/87
413 061900 CLOSE INTFIL DSPFIL OPRINT. 11/21/88
414 062000 STOP RUN. 10/01/87
062100 09/30/87
062200 ************************************************************************* 03/16/89
062300+ * 03/16/89
573CBI V2R1MO 910524 AS/400 COBOL/400 Source INTLIB/CSINT RCH38321 10/08/90 11:09:19 Page 17
STMT SEQRNR - A I B *.+.2++.+.3++.+.4++.+.5++.+.6++.+.7++.+.IDENTCN COPYNAME CGN DATE
421 064000+ THIS ROUTINE IS CALLED TO END THE TRANSACTIONS WITH THE * 03/16/89
064200+ TARGET PROGRAMS. * 03/16/89
064300+ * 03/16/89
064700************************************************************************* 03/16/89
064800+ 30 10/08/90
064900 DETACH-Routine. 09/30/87
415 065300 MOVE "ICF01 " TO PGM-DEV-NME. 09/30/87
416 065100 WRITE INTREC FORMAT IS "DETACH" 11/21/88
065200 TERMINAL IS PGM-DEV-NME. 09/30/87
417 065300 MOVE "ICF01 " TO PGM-DEV-NME. 09/30/87
418 065400 WRITE INTREC FORMAT IS "DETACH" 11/21/88
065500 TERMINAL IS PGM-DEV-NME. 09/30/87
065600 DETACH-EXIT. 09/30/87
065700 EXIT. 09/30/87
065800+ 30 09/30/87
065900************************************************************************* 03/16/89
066400+ * 03/16/89
066410+ THIS ROUTINE IS CALLED TO RELEASE THE PROGRAM DEVICES, END * 03/16/89
066420+ THE SESSIONS, AND END THE PROGRAM. * 03/16/89
066430+ * 03/16/89
066440************************************************************************* 03/16/89
066500+ 30 10/08/90
066600+ 30 09/30/87
419 066700 END-JOB. 09/30/87
420 066800 DROP "ICF00 " FROM INTFIL. 11/21/88
421 066900 DROP "ICF01 " FROM INTFIL. 11/21/88
422 065000 CLOSE INTFIL DSPFIL OPRINT. 11/21/88
423 065100 STOP RUN. 09/30/87

***** END OF SOURCE *****
573CBI V2R1MO 910524 AS/400 COBOL/400 Messages INTLIB/CSINT RCH38321 10/08/90 11:09:19 Page 18
STMT
* 25 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 004700
Message . . . . : No input fields found for format DETACH.
* 25 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 004700
Message . . . . : No input fields found for format DETACH.
* 25 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 004700
Message . . . . : No input fields found for format DETACH.
* 25 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 004700
Message . . . . : No input fields found for format DETACH.
* 25 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 004700
Message . . . . : No input fields found for format DETACH.
* 25 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 004700
Message . . . . : No input fields found for format DETACH.
* 103 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 005100
Message . . . . : No output fields found for format CIMENU.
* 103 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 005100
Message . . . . : No output fields found for format DTMMNU.
* 103 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 005100
Message . . . . : No output fields found for format DTMMNU.
* 103 MSGID: LBL0600 SEVERITY: 10 SEQRNR: 005100
Message . . . . : No output fields found for format DTMMNU.

***** END OF MESSAGES *****
Message Summary
Total Info(0-4) Warning(5-9) Error(20-29) Severe(30-39) Terminal(60-99) 11 0 11 0 0 0
11 Source records read . . . . . . . . : 629
Copy records read . . . . . . . . . . : 212
Copy members processed . . . . . . . : 2
Sequence errors . . . . . . . . . . : 0
Highest severity message issued . . . . : 10
LBL0901 0D Program CSDINT created in library INTLIB.

***** END OF COMPI LATION *****
Figure D-10 (Part 13 of 13). Source Program Example — CSDINT
COBOL/400 Target Program for a Two-Session Inquiry

The following describes the COBOL/400 target program for a two-session inquiry.

Program Files: The COBOL/400 two-session target program uses the following files:

CFILE  An ICF file used to send records to and receive records from the source program. It is done with the file-level INDARA DDS keyword, indicating a separate indicator area.

PFILE  A database file used to retrieve the record for the item requested from the source program.

QPRINT  An AS/400 printer file used to print records, both sent and received, as well as major and minor ICF return codes.

DDS Source: The DDS for the ICF file (CFILE) is illustrated in Figure D-11 on page D-34.

The DDS source for the database file (PFILE) is illustrated in Figure D-12 on page D-34.

ICF File Creation and Program Device Entry Definition: The command needed to create the ICF file is:

```plaintext
CRTICFF FILE(INTLIB/CFILE)
   SRCFILE(INTLIB/QINTSRC) SRCMBR(CFILE)
   ACQPGMDEV(RQSDEV)
   TEXT("TARGET ICF FILE FOR TWO SESSION PROGRAM")
```

The command needed to define the program device entry is:

```plaintext
OVRICFDEVE PGMDEV(RQSDEV)
   RMTLOCNAME(/c5197REQUESTER)
```

Program Explanation: The following explains the structure of the program example illustrated in Figure D-13 on page D-35. The ICF file used in the example is defined by the user, and uses externally described data formats (DDS). The reference letters in the example below correspond to those in the following program example.

1. This section defines the ICF file (CFILE) and the database file (PFILE) used in the program.

2. This section defines the error handling for the program. The CFILE routine first checks the major/minor return code to determine if the error is recoverable.

3. If any other error has occurred, the program prints a message saying that the program ended abnormally and then ends.

4. This routine opens all the files.

5. The DDS for the database file (PFILE) is illustrated in Figure D-12 on page D-34.

6. The program uses the requested number received from the source program to access the record from the database. The information retrieved from the database file (PFILE) is moved to the work area for the ICF file. A write operation is issued to the program device using record format SNDPART. The write operation sends the requested information back to the source program.

7. If the requested number is not found, a fail indication is sent to the requesting program using a write operation with a fail operation.

8. If an error occurs on the write operation, control passes to section 2.

9. If no error occurs on the write, control goes back to section 4.

10. A read operation is issued to the program device.

11. If a detach indication is received, the program goes to section 8 to end the program. When a detach is received, indicator 44 is set on, as defined by the RCVDETACH keyword in the DDS for the ICF file.

12. This routine is called to end the program.

The following message is written to the printer file:

```plaintext
CTDINT HAS COMPLETED NORMALLY
```

The files are closed. The program device is automatically released as a result of the close operation and the program ends.
D-34
Intrasystem Communications Programming V4R1

Figure D-11. DDS Source for a Two-Session Target Program Using CFILE

```plaintext
5714PW1 RO1300 880301  SEU SOURCE LISTING  10/16/87 07:43:14  PAGE 1
SOURCE FILE . . . . . . QINTSRC/INTLIB
MEMBER . . . . . . . . . PFILE
SEQNR+...+ 1 2 3 4 5 6 7 8 9 0
A******************************************************************************
A  ICF FILE
A  USED IN TARGET TWO SESSION PROGRAM
A  *
A******************************************************************************
A  ININDA
A  05  RQSWRT
A  10  ALWRT
A  ININDT(10 '10 END TRANS.')
A  15  EOS
A  20  FAIL
A  ININDT(20 '20 F ABORT ST')
A  RCVFAIL(25 'RECEIVED FAIL')
A  30  DETACH
A  ININDT(30 '30-DETACH TGT')
A  RCVDETACH(44 'RCV DETACH')
A  RCVTRNRND(40 'END OF TRN')
A  R SNDPART
A  INVITE
A  RECTYP 1
A  ITEMNO 6
A  EDATA 130
A  FILL1 13
A  R RCVPART
A  REC102 6
A  PARTDS 80
A  FILL4 64
END OF SOURCE

Figure D-12. DDS Source for a Two-Session Target Program Using PFILE

```
Appendix D. Program Examples

Figure D-13 (Part 1 of 4). Target Program Example — CTDINT (User-Defined Formats)
+000004+THE KEY DEFINITIONS FOR RECORD FORMAT DBREC \<ALL-FMTS
+000005+ NUMBER NAME RETRIEVAL TYPE ALSEQ \<ALL-FMTS
+000006+ 0001 DBSEQ ASCENDING AND NO \<ALL-FMTS
27 +000007 05 DBREC REDEFINES PPFILE-RECORD. \<ALL-FMTS
28 +000008 06 RECCUS PIC X(1). \<ALL-FMTS
29 +000009 06 DBSEQ PIC X(6). \<ALL-FMTS
30 +000010 06 DBDATA PIC X(130). \<ALL-FMTS
31 +000011 06 DBFILL PIC X(13). \<ALL-FMTS
32 004100 FD CFILE \<ALL-FMTS
33 004200 LABEL RECORDS ARE STANDARD. \<ALL-FMTS
34 004300 01 ICFREC \<ALL-FMTS
35 004400 COPY DDS-ALL-FORMATS-I-O OF CFILE. \<ALL-FMTS
36 +000001 05 CFILE-RECORD PIC X(150). \<ALL-FMTS
+000002+ INPUT FORMAT:SNDPART FROM FILE CFILE OF LIBRARY INTLIB \<ALL-FMTS
+000003+ OUTPUT FORMAT:SNDPART FROM FILE CFILE OF LIBRARY INTLIB \<ALL-FMTS
+000010+ INPUT FORMAT:SNDPART FROM FILE CFILE OF LIBRARY INTLIB \<ALL-FMTS
37 +000004 05 SNDPART-I REDEFINES CFILE-RECORD. \<ALL-FMTS
38 +000005 06 RECYPIC X(1). \<ALL-FMTS
39 +000006 06 ITEMNO PIC X(6). \<ALL-FMTS
40 +000007 06 EDATA PIC X(130). \<ALL-FMTS
41 +000008 06 FILL PIC X(13). \<ALL-FMTS
+000009+ OUTPUT FORMAT:SNDPART FROM FILE CFILE OF LIBRARY INTLIB \<ALL-FMTS
42 +000011 05 SNDPART-O REDEFINES CFILE-RECORD. \<ALL-FMTS
43 +000012 06 RECYPIC X(6). \<ALL-FMTS
44 +000013 06 ITEMNO PIC X(6). \<ALL-FMTS
45 +000014 06 EDATA PIC X(130). \<ALL-FMTS
46 +000015 06 FILL PIC X(13). \<ALL-FMTS
+000016+ INPUT FORMAT:RCVPART FROM FILE CFILE OF LIBRARY INTLIB \<ALL-FMTS
+000017+ OUTPUT FORMAT:RCVPART FROM FILE CFILE OF LIBRARY INTLIB \<ALL-FMTS
47 +000018 05 RCVPART-I REDEFINES CFILE-RECORD. \<ALL-FMTS
48 +000019 06 RECIPIC X(6). \<ALL-FMTS
49 +000020 06 PARTDS PIC X(80). \<ALL-FMTS
50 +000021 06 FILL PIC X(64). \<ALL-FMTS
+000022+ OUTPUT FORMAT:RCVPART FROM FILE CFILE OF LIBRARY INTLIB \<ALL-FMTS
+000023+ OUTPUT FORMAT:RCVPART FROM FILE CFILE OF LIBRARY INTLIB \<ALL-FMTS
51 +000024 05 RCVPART-O REDEFINES CFILE-RECORD. \<ALL-FMTS
52 +000025 06 RECIPIC X(6). \<ALL-FMTS
53 +000026 06 PARTDS PIC X(80). \<ALL-FMTS
54 +000027 06 FILL PIC X(64). \<ALL-FMTS
55 004500 FD QPRINT \<ALL-FMTS
56 004600 LABEL RECORDS ARE OMITTED. \<ALL-FMTS
57 004700 01 PRINTREC \<ALL-FMTS
58 004800 05 RC PIC 9999. \<ALL-FMTS
59 004900 05 ERRMSG PIC X(128). \<ALL-FMTS
60 005000 WORKING-STORAGE SECTION. \<ALL-FMTS
61 005100 77 MAJ-MIN-SAY PIC X(4). \<ALL-FMTS
62 005200 77 STATUS-IND PIC X(2). \<ALL-FMTS
63 005300 77 INDON PIC 1 VALUE '1'. \<ALL-FMTS
64 005400 77 INDOFF PIC 1 VALUE '0'. \<ALL-FMTS
65 005500 77 LEN PIC 9(10)V9(5) COMP \<ALL-FMTS
66 005600 VALUE 0. \<ALL-FMTS
67 005700 77 CMD02 PIC X(31) \<ALL-FMTS
68 005800 VALUE 'CPYF HEXDUMP +LIST PRTFMT(+HEX)'. \<ALL-FMTS
69 005900 01 CMN-INDIC-AREA \<ALL-FMTS
70 006000+ ALLOW WRITE (ALWRT) INDICATOR \<ALL-FMTS
71 006100 05 INIC PIC 1 INDIC 10. \<ALL-FMTS
72 006200 88 INIC-ON VALUE '1'. \<ALL-FMTS
73 006300 88 INIC-OFF VALUE '0'. \<ALL-FMTS
74 006400+ FAIL (FAIL) INDICATOR \<ALL-FMTS
75 006500 05 INIC PIC 1 INDIC 20. \<ALL-FMTS
76 006600 88 INIC-ON VALUE '1'. \<ALL-FMTS
77 006700 88 INIC-OFF VALUE '0'. \<ALL-FMTS
78 006800+ RECEIVE FAIL (RCVFAL) INDICATOR \<ALL-FMTS
79 006900 05 IN25 PIC 1 INDIC 25. \<ALL-FMTS
80 007000 88 IN25-ON VALUE '1'. \<ALL-FMTS
81 007100 88 IN25-OFF VALUE '0'. \<ALL-FMTS
82 007200+ RECEIVE TURNAROUND (RCVRTNRD) INDICATOR \<ALL-FMTS

Figure D-13 (Part 2 of 4). Target Program Example — CTDINT (User-Defined Formats)
Appendix D. Program Examples

Figure D-13 (Part 3 of 4). Target Program Example — CTDINT (User-Defined Formats)
Figure D-13 (Part 4 of 4). Target Program Example — CTDINT (User-Defined Formats)
RPG/400 Source Program for a Two-Session Inquiry

The following describes an RPG/400 source program for a two-session inquiry.

**Program Files:** The RPG/400 two-session source program uses the following files:

- **INTFIL**: An ICF file used to send records to and receive records from the target program.
- **DSPFIL**: A display file used to enter requests to be sent to the target program.
- **QPRINT**: An AS/400 printer file used to print records, both sent and received, as well as major and minor ICF return codes.

**DDS Source:** The DDS for the ICF file (INTFIL) is illustrated in Figure D-14.
The DDS source file for the display file (DSPFIL) is shown in Figure D-15.

Figure D-15 (Part 1 of 2). DDS Source for Source Program Two-Session Inquiry Using DSPFIL
**Configuration:** The following command is needed to create the intrasystem communications device associated with the ICF file:

```
CRTDEVINTR DEVD(INTRADEV) RMTLOCNAME(INTRARMT) ONLINE(/c5197NO) TEXT("THIS IS AN INTRASYSTEM DEVICE DESCRIPTION")
```

**ICF File Creation and Program Device Entry Definition:**

The command needed to create the ICF file is:

```
CRTICFF FILE(INTLIB/INTFIL) SRCFILE(INTLIB/QINTSRC) SRCMBR(INTFIL) ACQPGMDEV(*NONE) MAXPGMDEV(2) WAITRCD(30) TEXT("SOURCE ICF FILE FOR TWO SESSION PROGRAM")
```

It is not necessary to add a communications entry to the subsystem since the system automatically defines an entry for the device created above at run time. However, if you decide to have one, the following is an example:

```
ADDCMNE SBSD(QCMN) DEV(INTRADEV)
```

The commands needed to define the two program device entries are:

```
OVRICFDEVE PGMDEV(ICF/zerodot/zerodot) RMTLOCNAME(INTRARMT) FMTSLT(*RECID) OVRICFDEVE PGMDEV(ICF/zerodot1) RMTLOCNAME(INTRARMT) FMTSLT(*RECID)
```

The following is an example of a CL program that might be used to run the source program shown in the example above:

```
RSDINTCL: PGM PARM(&RMT1 &RMT2) DCL VAR(&RMT1) TYPE(/c5197CHAR) LEN(8) DCL VAR(&RMT2) TYPE(/c5197CHAR) LEN(8) CHGJOB OUTQ(INTLIB/INTOUTQ) LOG(4 /zerodot/zerodot /c5197SECLVL) LOGCLPGM(/c5197YES) OVRICFDEVE PGMDEV(ICF/zerodot/zerodot) RMTLOCNAME(&RMT1) FMTSLT(*RECID) OVRICFDEVE PGMDEV(ICF/zerodot1) RMTLOCNAME(&RMT2) FMTSLT(*RECID) CALL INTLIB/RSDINT ENDRSDINTCL: ENDPGM
```
The following is an example of a CL program that might be used as the target program that your source program evokes (which calls the program RTDINT shown in the example):

RTDINTCL: PGM
   CHGJOB OUTQ(INTLIB/INTOUTQ)
   LOG(4 00 +SECLVL)
   LOGCPLGM(+YES)
   ADDLIBL INTLIB
   OVRICFDEVE PGMDEV(RQSDEV)
   RMTLOCNAME(+REQUESTER)
   CALL INTLIB/RTDINT
   RMLVLIBLE INTLIB
ENDRTDINTCL: ENDPGM

Program Explanation: The following explains the structure of the program example illustrated in Figure D-16 on page D-44. The ICF file used in the example is defined by the user, and uses externally described data formats (DDS). The reference numbers in the explanation below correspond to the numbers in the following program example.

The ICF file used in the example is externally described.

All output operations to the ICF file in the example are done using the WRITE statement.

1. The file specifications define the ICF file (INTFIL) and the display file (DSPFIL) used in the program.
   - INTFIL is the ICF file used to send records to and receive records from each of the two target programs.
   - DSPFIL is the display file used to receive user’s requests and to report the information received based on the request.

The files used in the program are opened at the beginning of the RPG/400 cycle.

Note: The continuation lines on the file specification define the following:
   - The data structure names, IOFB and IODS, used for the feedback area (INFDS) for INTFIL and DSPFIL respectively.
   - The number of program devices that can be attached to the files (two for INTFIL).
   - The program device name in the CMID field to which it issues the I/O operation.

2. The file information data structure (IOFB) is provided to receive the I/O feedback area following an ICF file I/O operation.
   - For the display file, the file information data structure (IODS) is used by the program to determine the record format used for the last display file I/O operation. The field name referred to in the program is RECID, found in positions 261 through 268 of the feedback area.
   - The two ICF program devices used by the program are explicitly acquired.
   - The work station is implicitly acquired when the DSPFIL file opens.
   - Also, the evoke requests are issued to the remote systems by the subroutine EVKSR in section 13.

3. When control returns from the EVKSR subroutine, the main menu (record format CIMENU) is written to the work station.

4. A read operation is issued to the display program device and the program waits for an input request from the user. When a record is returned, the last record format used (as specified in the RECID field in the I/O feedback area) is checked. The program branches to the appropriate routine according to the value in RECID.

5. The request entered by the user from the main menu (CIMENU) is checked. If indicator 99 is set to 1, indicating that the operator pressed function key 1, the two transactions and sessions end and the program ends. If the operator entered option 1, the program writes the item inquiry menu (ITMIMNU) to the work station and returns to the read to the display program device section (4).

6. If the option is not 1, the Buyer Inquiry menu (DTLMNU) is written to the work station and control is passed to section 4.

The item number requested by the user from the Order Inquiry Display (record format ITMIMNU) is processed here. If function key 1 is pressed (indicator 99), control passes to the I/O operation error section (section 12), the two transactions and sessions are ended, and the program ends. If function key 2 is pressed, the inquiry request is canceled, the main menu (CIMENU) is written to the work station, and the program returns to section 4.

The item number is read from the work station and then the request is sent to the target program on program device ICF01.

The request is sent to the appropriate target program by writing data to the program device using format ITMREQ. The INVITE keyword is specified as part of the ITMREQ format to give the target program permission to send.

A timer is issued for 30 seconds before the read operation. This is provided to allow the local program to have a time-out when no response is received from the target program.

A read-from-invited-program-devices operation is issued to the invited program device to receive the response to the inquiry. The operation is interpreted as a read-from-invited-program-devices because the program device name field (CMID) is blank. Indicator 89 is set on after I/O operation, if the operation does
not complete. The subroutine ERRCHK in section 14 gets control, and further checks are made.

The return codes are checked after an I/O request. If there are any errors, control is passed to section 12. If not, the program returns to section 4.

The information received from the target program is processed. If the information received is a fail indication, it means the requested item number was not found and the request is not valid. A new Item Inquiry menu (ITMMNU) is written to the work station, and control goes to section 4.

The program then performs the calculations to set the quantity fields and writes the result to the requesting work station using record format ITMSC2.

The program then returns to section 4.

This section processes the user requests for additional information (record format ITMSC2). If function key 2 (indicator 98) is pressed, the main menu (record format CIMENU) writes to the work station and control goes to section 4.

If the Enter key is pressed, the profit and loss figures are calculated. Those values are then written to the work station using format ITMSC3 (item inquiry work station 3). The program then returns to section 4. If function key 1 (indicator 99) was pressed, control goes to section 12.

If function key 3 (indicator 97) is pressed, the Order Inquiry menu (ITMMNU) is written to the work station and the program returns to section 4.

This section processes inquiry read requests from the Buyer menu (DTLMNU). If function key 2 (indicator 98) is pressed, the main menu (CIMENU) is written to the work station and the program returns to section 4. If function key 1 (indicator 99) is pressed, control goes to section 12.

The buyer inquiry request is sent to the target program by writing data to the program device (ICF00) using format DLTREQ. The INVITE keyword is specified as part of the DLTREQ format to give the target program permission to send.

A read operation is issued to the invited program device to receive the response to the inquiry. This is accomplished by blanking out CMID. Indicator 88 is set on if the I/O operation did not complete.

**Note:** A timer operation is issued before the above read is issued to ensure that the operation will finish even if the target program is unable to respond.

If the information received is a fail indication (indicator 25) from the target program, it means the requested item was not found and the request is not valid. The main menu (record format CIMENU) is written to the work station. The program then returns to section 4.

The return codes (or indicators) are checked after an I/O request. If there are any errors, control is passed to section 12.

The information supplied by the target program in response to a request for a buyer detail is processed.

The detail information is written to the work station using record format DTLSCR.

The program then returns to section 4.

Control is passed here if the buyer detail record format (DLTSCR) is displayed. If function key 1 (indicator 99) is pressed, control goes to section 12. If function key 2 (indicator 98) is pressed, the main menu (CIMENU) is written to the work station and control is returned to section 4.

If the record format name is not found on a read operation, an error message prints. If an error occurs on any ICF operation, control is passed here and an error message is printed containing the program device and error that occurred.

For each of the two sessions, the transaction is ended by issuing a detach request to the appropriate program device using format DETACH, and the session is ended by the release operation. The last record indicator is turned on to end the program. The ICF file is implicitly closed at the end of the RPG/400 cycle.

The EVKSR subroutine in this section builds the evoke requests to send to the remote programs. Because the DDS keyword for the record format only specifies the field identifiers with the record, this code moves the literal value RTDINTCL to the field PGMID, and INTLIB to the field LIB.

When the program start request is received at the remote program, INTLIB is searched for RTDINTCL and that program is then started. RTDINTCL is a CL program that contains CL statements as illustrated on D-41.

The subroutine ERRCHK is called when the read operation to the program device does not complete. The indication that the timer has ended is checked (RC=0310) and if it is set, a message is displayed to the user. The message asks whether you want to try the read operation again or end the job. In this example, the time interval is specified in section 9.

The subroutine *PSSR is called if there are I/O operation errors that are not handled by the subroutine ERRCHK in section 14. It checks to see whether the program device is already acquired when an acquire operation is requested and if it is, the second acquire is ignored. Otherwise, the program ends.
Figure D-16 (Part 1 of 13). Source Program Example — RSDINT

Record format(s): library INTLIB file INTFIL.

EXTERNAL FORMAT ITMRSP RPG NAME ITMRSP
EXTERNAL FORMAT DTLRSP RPG NAME DTLRSP
EXTERNAL FORMAT DETACH RPG NAME DETACH
EXTERNAL FORMAT EOS RPG NAME EOS
EXTERNAL FORMAT EVKREQ RPG NAME EVKREQ

Source Listing

100 H 1 10/13/87
200 H**************************************************** 10/13/87
300 H THIS PROGRAM Allocates TWO SESSIONS AS FOLLOWS: 11/15/88
400 H 'ICF00' TO INQUIRE ABOUT A BUYER'S CREDIT STANDING 11/15/88
500 H BEFORE AN ORDER IS PROCESSED. 11/15/88
600 H 'ICF01' TO INQUIRE ABOUT THE AVAILABILITY OF AN ITEM 11/15/88
700 H BEING ORDERED (ITEM 000001 THRU 999999). 11/15/88
800 H A DISPLAY DEVICE IS USED TO ENTER THE REQUEST ( USING A 10/13/87
900 H BUYER AND AN ITEM MENU THAT IS SENT TO THE TARGET 10/05/90
1000 H PROGRAM. 10/05/90
1100 Hffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
Figure D-16 (Part 2 of 13). Source Program Example — RSDINT
Figure D-16 (Part 3 of 13). Source Program Example — RSDINT
**Figure D-16 (Part 4 of 13). Source Program Example — RSDINT**

```
9900 C****************************************************************************** 10/13/87
10000 C* DETERMINE USER'S REQUEST * 03/17/89
10100 C* A READ TO THE DISPLAY DEVICE IS ISSUED TO RECEIVE THE * 03/17/89
10200 C* USER'S REQUEST, THE TYPE OF REQUEST MADE IS BASED ON THE * 03/17/89
10300 C* DISPLAY FORMAT CURRENTLY ON THE SCREEN. THE RECORD FORMAT * 03/17/89
10400 C* NAME IS EXTRACTED FROM THE I/O FEEDBACK AREA AND USED TO * 03/17/89
10500 C* DETERMINE WHAT ACTION SHOULD BE TAKEN NEXT. * 03/17/89
10600 C****************************************************************************** 10/13/87
10700 C****************************************************************************** 10/13/87
10800 C****************************************************************************** 10/13/87
10900 C****************************************************************************** 10/13/87
11000 EXEC INDLB/RSDINT 10/05/90 16:12:28 Page 6
11100 C* RELOAD INDEX * 10/13/87
11200 C****************************************************************************** 10/13/87
11300 C****************************************************************************** 10/13/87
11400 C****************************************************************************** 10/13/87
11500 C****************************************************************************** 10/13/87
11600 C****************************************************************************** 10/13/87
11700 C****************************************************************************** 10/13/87
11800 C****************************************************************************** 10/13/87
11900 C****************************************************************************** 10/13/87
12000 C****************************************************************************** 10/13/87
12100 C****************************************************************************** 10/13/87
12200 C****************************************************************************** 10/13/87
12300 C****************************************************************************** 10/13/87
12400 C****************************************************************************** 10/13/87
12500 C****************************************************************************** 10/13/87
12600 C****************************************************************************** 10/13/87
12700 C****************************************************************************** 10/13/87
12800 C****************************************************************************** 10/13/87
12900 C****************************************************************************** 10/13/87
13000 C****************************************************************************** 10/13/87
13100 C****************************************************************************** 10/13/87
13200 C****************************************************************************** 10/13/87
13300 C****************************************************************************** 10/13/87
13400 C****************************************************************************** 10/13/87
13500 C****************************************************************************** 10/13/87
13600 C****************************************************************************** 10/13/87
13700 C****************************************************************************** 10/13/87
13800 C****************************************************************************** 10/13/87
13900 C****************************************************************************** 10/13/87
14000 C****************************************************************************** 10/13/87
14100 C****************************************************************************** 10/13/87
14200 C****************************************************************************** 10/13/87
14300 C****************************************************************************** 10/13/87
14400 C****************************************************************************** 10/13/87
14500 C****************************************************************************** 10/13/87
14600 C****************************************************************************** 10/13/87
14700 C****************************************************************************** 10/13/87
14800 C****************************************************************************** 10/13/87
14900 C****************************************************************************** 10/13/87
15000 C****************************************************************************** 10/13/87
15100 C****************************************************************************** 10/13/87
15200 C****************************************************************************** 10/13/87
15300 C****************************************************************************** 10/13/87
15400 C****************************************************************************** 10/13/87
15500 C****************************************************************************** 10/13/87
15600 C****************************************************************************** 10/13/87
15700 C****************************************************************************** 10/13/87
15800 C****************************************************************************** 10/13/87
15900 C****************************************************************************** 10/13/87
16000 C****************************************************************************** 10/13/87
16100 C****************************************************************************** 10/13/87
16200 C****************************************************************************** 10/13/87
16300 C****************************************************************************** 10/13/87
16400 C****************************************************************************** 10/13/87
```
Figure D-16 (Part 5 of 13). Source Program Example — RSDINT
Appendix D. Program Examples

Figure D-16 (Part 6 of 13). Source Program Example — RSDINT
IF AN ERROR OCCURS, THE ERROR IS PRINTED AND THE JOB IS ENDED.

---

**Figure D-16 (Part 7 of 13). Source Program Example — RSDINT**

---

**Sequence Ind Do Last Page Program**

---

**Notes:**

- If an error occurs, the error is printed and the job is ended.
- Process buyer information.
- Display the buyer information.
- Exit the job.
- "Enter" will bring up the buyer inquiry menu.
- Display the buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
- Display the main menu and buyer inquiry menu.
- Display the main menu and buyer information.
Appendix D. Program Examples

D-51

Figure D-16 (Part 8 of 13). Source Program Example — RSDINT

SEQUENCE IND DO LAST PAGE PROGRAM

Number UPDATE LINE ID

43500 C GOTO CHKAGN ASK AGAIN 001 10/13/87
43600 C END 001 10/13/87
43700 C *IN25 IFEQ '1' RECEIVE FAIL 001 11/16/88
43800 C WRITEMENU MAIN MENU 001 11/16/88
43900 C GOTO READQR 001 11/16/88
44000 C END 001 11/16/88
44100 C GOTO ERROR ABEND 10/13/87
44200 C ENDSR 10/13/87

Appendix D. Program Examples

D-51

Figure D-16 (Part 8 of 13). Source Program Example — RSDINT
Figure D-16 (Part 9 of 13). Source Program Example — RSDINT
**Appendix D. Program Examples**

**Figure D-16 (Part 10 of 13). Source Program Example — RSDINT**

```
1000000 OUTPUT FIELDS FOR RECORD ITMSC3 FILE DSPFIL FORMAT ITMSC3.
1000000 ITEM INQUIRY SCREEN 3
1000001 SLM 9 ZONE 9,2
1000002 SLSY 20 ZONE 11,2
1000003 CSTM 29 ZONE 9,2
1000004 CSTY 40 ZONE 11,2
1000005 PROFIT 45 ZONE 5,2
1000006 LOSTS 54 ZONE 9,2
2000000 OUTPUT FIELDS FOR RECORD TIMOUT FILE DSPFIL FORMAT TIMOUT.
2000000 TIME OUT SCREEN

**** **END OF SOURCE ****

**Additional Diagnostic Messages**

* 7899 3000 RPG PROVIDES SEPARATE INDICATOR AREA FOR FILE INTFIL.

573BG1 V2RMD 910524 10/05/90 16:12:28 Page 14

**Cross Reference**

File and Record References:

<table>
<thead>
<tr>
<th>Field</th>
<th>Attr</th>
<th>References (D=DEFINED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2</td>
<td></td>
<td>FILE/RCD WORKSTN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CIMENU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DTMENU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DTLSR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EVKREQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITMREQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITMRSP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIMOUT</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>INFIL WORKSTN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATIMOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VRMNT</td>
</tr>
</tbody>
</table>

Field References:

<table>
<thead>
<tr>
<th>Field</th>
<th>Attr</th>
<th>References (M=MODIFIED D=DEFINED)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>+INZS A(1) 18200 32100 43700</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+IN7 A(1) 10000001 J0000001 K0000001 L0000001 M0000001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N0000001 0000001 25800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+IN98 A(1) 1000002 J000002 K000002 L000002 M000002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N000002 000002 16800 25400 30700</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+IN99 A(1) 1000003 J000003 K000003 L000003 M000003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N000003 000003 13500 16700 25300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+PSR A(1) 36100 36000 13500 16700 25300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ABORT TAG 38600D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHKAGN 42600D 43500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMD 6700D 9300M 9500M 17200M 17900M</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31200M 31800M 38200M 38400M 47100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CNAME A(5) 34100M X000000D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CSTM P(9,2) 27500M 1000000D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CSTMT P(9,2) A000014D 26000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>573BG1 V2RMD 910524 IBM AS/400 RPG/400</td>
</tr>
</tbody>
</table>

**Figure D-16 (Part 10 of 13). Source Program Example — RSDINT**

Appendix D. Program Examples  D-53
DTLIN TAG 11800 305000  
DLRTN TAG 11900 359000  
* 7031 DTOUT TAG 33900D  
END TAG 13500 16700 25300 30600 36000  
37800 381000 41000  
ENPSR ENPSR 45800 460000  
* 7031 ENTRY TAG 9000D  
ERR Z(5,3) 64000 45800  
ERCHK BEGSR 18100 32000 424000  
ERROR TAG 17500 18700 31400 32700 379000  
44100  
EVTL TAG 31100 32600  
EVKSRS BEGSR 9400 9600 404000  
* 7031 FILL1 A(240) 5900D  
* 7031 FILL2 A(145) 6100D  
* 7031 FILL3 A(240) 6500D  
* 7031 FILL4 A(145) 7200D  
* 7031 FILL1 A(56)  
* 7031 FILL2 A(57) 8000011D  
PFTNM A(10) 6600D 46900  
FORCE TAG 38000 43300  
IFDPT P(3,3) 8000010D 34700  
ELS A(DS(415)) 4400 58000D  
IFB DS(DS(415)) 3900 62000  
ITEMNO P(6,3) A0000020 F00001D 0000001D 26050 5000001D  
ITMGTN TAG 11500 166000 18600  
* 7031 LMTOUT TAG 20400D  
ITMNTEXT TAG 11600 11700 25200D  
LIR A(10) 40600M 408000M 0000020D  
* 7031 LOC A(8) 6300D  
* 7031 LOS P(9,2) A0000170  
LSTPST P(9,2) 272000 1000060  
* 7031 MAIN TAG 9700D  
MAJCOD A(2) 70000 46700 47700  
MMERR EXCPT 38000 46200  
OPTION A(1) 1000004D 13600  
PMID A(10) 40500M 407000M 000001D  
* 7031 PRO P(5,2) A000018D  
PROFIT P(5,2) 27600M 10000050  
PROPM P(9,2) 26800D 26900D 27100D 271000M 27600  
PRO1 P(7,2) A0000090 218000 27200  
PRO3 P(7,0) A000110D 219000  
PRI P(7,2) 21800M Z000007D  
PRI P(7,0) 21900D Z000008D  
QAVA1 P(7,0) 20900D 210000 211000M 212000Z 0000020D  
QYB P(7,0) 21600M Z000005D  
QYBD P(7,0) A0000070 212000 21600  
QYH P(7,0) 21500M Z0000030  
QYLIST P(7,0) A000004D 27200  
QYO P(7,0) 21400M Z0000040  
QYOH P(7,0) A000005D 21000 21500  
QYOH P(7,0) A000006D 21100 21400  
READRQ TAG 11100D 12100 14100 17000 18400  
20700 22200 25600 26100 26600  
27900 30900 32400 34900 36300  
36600 43900  
* 7031 RECCUS A(1) B00001D  
RECER EXCPT 37700 47200  
RECERR TAG 18800 32800 376000  
RECID A(8) 6000D 11400 11500 11600 11700  
11800 11900 25900 26400  
RECIPD A(8) 7100D 18800 32800 47900  
* 7031 RECINM A(1) A00001D  
RETURN A(6) 457000 459000 46000  
SLSM P(9,2) 27300M 100001D  
SLSM P(9,2) A000012D 26800 27000 27100 27300  
SLSY P(11,2) A000013D 27400  
SLSY P(11,2) 27400M 1000002D  
TIMRSP A(1) 000004D 42900 43000 43100  
TRYBB TAG 31500 42900  
TRYBB TAG 17600 43000  
UFR P(5,2) 22000M Z000009D  
UFRT P(5,2) A000011D 22000  
UNITQ A(2) A000008D 21700  
UNT A(2) 21700M Z0000060  

Figure D-16 (Part 11 of 13). Source Program Example — RSDINT

D-54 Intrsystem Communications Programming V4R1
Figure D-16 (Part 12 of 13). Source Program Example — RSDINT

Message Summary

* 7031 XITMIN TAG 1730DD
  *BLANK LITERAL 04050 04600 04050 04600
  * LITERAL 17900 31800
  * LITERAL 45700
  *CANCE LITERAL 49500
  'CIMENU' LITERAL 11400
  'DLIN MU' LITERAL 11800
  'DLRSP' LITERAL 32800
  'DLTSP' LITERAL 11900
  'ICF00' LITERAL 9100 9300 31200 38200 38700
  'ICF01' LITERAL 9200 9500 17200 38400 38800
  'INTLIB' LITERAL 40800
  'ITMMNU' LITERAL 11500
  'ITMRSP' LITERAL 10800
  'ITMSC2' LITERAL 11600 25900
  5738RG1 V2R1M 910524 IBM A5/400 RPG/400 INTLIB/RSDINT 10/05/90 16:12:28 Page 17
  'ITMSC3' LITERAL 11700 26400
  'RTDINTCL' LITERAL 40700
  '0300' LITERAL 18600 32600
  '0310' LITERAL 42500
  '04' LITERAL 17500 18700 31400 32700 41000
  '1' LITERAL 13500 13600 16700 16800 18200
  25300 25400 25800 30600 30700
  32100 36000 36100 42900 43000
  43700
  '2' LITERAL 43100
  0 LITERAL 20900 27000
  000000 LITERAL 20500
  01285 LITERAL 45800
  100 LITERAL 26900

Indicator References:
  INDICATOR REFERENCES (M=MODIFIED D=DEFINED)
  +IN 1000001 1000002 1000003 J000001 J000002 J000003
  K000001 K000002 K000003 L000001 L000002 L000003
  M000001 M000002 M000003 N000001 N000002 N000003
  0000001 0000002 0000003 13500 16700 16800
  18200 25300 25400 25800 30600 30700
  32100 36000 36100 42900 43000
  43700
  LR 39000M
  0A 46000 48101
  + 7031 10 18000M 31900M
  25 18200 32100 43700
  46 27000M 27100
  + 7031 66 32200M
  + 7031 86 38700M 38800M 42800M
  + 7031 87 11300M
  88 11200M 31600M 31900M 32000 42900
  11200M 17700M 18000M 18100 43000
  + 7031 90 97 1000001 J000001 K000001 L000001 M000001 N000001
  0000001 25800
  98 1000002 J000002 K000002 L000002 M000002 N000002
  0000002 16800 25400 30700 36100
  99 1000003 J000003 K000003 L000003 M000003 N000003
  0000003 13500 16700 25300 30600 36000
  ** ** ** ** END OF CROSS REFERENCE ** ** ** **
  5738RG1 V2R1M 910524 IBM A5/400 RPG/400 INTLIB/RSDINT 10/05/90 16:12:28 Page 18
  ** ** ** ** END OF MESSAGE SUMMARY ** ** ** **
  * QRG0103 Severity: 00 Number: 1
  Message . . . . : The RPG provides Separate-Indicator area for file.
  * QRG7011 Severity: 00 Number: 23
  Message . . . . : The Name or indicator is not referenced.
  * QRG7089 Severity: 00 Number: 1
  Message . . . . : The Name or indicator is not referenced.
  ** ** ** ** END OF MESSAGE SUMMARY ** ** ** **

Figure D-16 (Part 12 of 13). Source Program Example — RSDINT
RPG/400 Target Program for a Two-Session Inquiry

The following describes an RPG/400 target program for a two-session inquiry.

Program Files: The RPG/400 two-session target program uses the following files:

- **CFILE**: An ICF file used to send records to and receive records from the source program.
- **PFILE**: A database file used to retrieve the requested information to send to the source program.
- **QPRINT**: An AS/400 printer file used to print records, both sent and received, as well as major and minor ICF return codes.

**DDS Source**: The DDS source for the ICF file (CFILE) is illustrated in Figure D-17.

---

```
A INDIRA
A 05 RQSWRT
A 10 ALWRT
A INDTXT(10 '10 END TRANS.')
A 15 EOS
A 20 FAIL
A INDTXT(20 '20 F ABORT ST')
A 30 RCVFAIL(25 'RECEIVED FAIL')
A 30 DETACH
A INDTXT(30 'DETACH TGT')
A 40 RCVDETACH(40 'RECV DETACH')
A R SNDPART
A INVITE
A RECYP 1
A ITEMNO 6
A IDATA 130
A FILL1 13
A R RCVPART
A RECID 6
A PARTDS 80
A FILL4 64
```

Figure D-17. DDS Source for an ICF File Used by a Target Program
The DDS for the database file (PFILE) is illustrated in Figure D-18.

ICF File Creation and Program Device Entry Definition:
The command needed to create the ICF file is:

```plaintext
CRTICFF FILE(INTLIB/CFILE)
SRCFILE(INTLIB/QINTSRC)
SRCMBR(CFILE)
ACQPGMDEV(RQSDEV)
TEXT("TARGET ICF FILE FOR TWO SESSION PROGRAM")
```

The command needed to define the program device entry is:

```plaintext
OVRICFDEVE PGMDEV(RQSDEV)
RMTLOCNAME(+REQUESTER)
```

Program Explanation: The following explains the structure
of the program example illustrated in Figure D-19 on page D-58. The ICF file used in the example is defined by the user, and uses externally described data formats. The reference numbers in the explanation below correspond to the numbers in the program example.

All output operations to the ICF file in the example are done using the write operation.

1. The file specification defines the files used in the program.
   CFILE is the ICF file used to send records to and receive records from the source program. The files used in the program are implicitly opened at the beginning of the RPG/400 cycle when the program starts.
   
   **Note:** The continuation lines on the file specification for CFILE define the data structure name; for example, FEEDBK for the feedback area (INFDS). FEEDBK contains the following information, which is used to monitor for error conditions after an I/O operation is issued to CFILE:
   - Record format name (FMTNM)
   - Program device name (PGMDEV)
   - Major/minor return code (MAJMIN)

2. A read operation is issued to the program device to receive an inquiry request from the source program. If an error occurs on the read operation (a major code greater than 03), control passes to the error section (section 5).

3. If a detach indication is received, control goes to section 6 of the program. Otherwise, the program goes to section 3. When a detach is received, indicator 44 is set on, as defined by the RCVDETACH DDS keyword in the ICF file.

4. If an error occurs (a major return code greater than 03 is returned from the read operation), the program goes to section 5. Otherwise, the program goes to section 4.

5. The program also tests to see whether the receive detach indicator (indicator 44) is set. If it is, the program goes to section 6.

6. The program uses the requested number received from the source program to access the record from the database. The information retrieved from the database file (PFILE) is moved into the work area for the ICF file. A write operation is issued to the ICF program device using record format SNDPART. The write operation sends the requested information back to the source program.

7. If the requested number is not found, a fail indication is sent to the remote program using a write operation in combination with a fail.

8. If an error occurs on the write operation (a major return code greater than 03), control passes to section 5.

9. If no error occurs on the write operation, the program returns to section 2.

10. When an error in an I/O operation is detected, an EXCPT operation is issued to print an error message saying that an error has occurred on the ICF file. The major/minor return code is also printed.

The program then goes to section 6.

11. Control passes to this section whenever the program has detected a communication error or has received a detach indication from the source program. The last record indicator is set on, which ends the program. CFILE is implicitly closed.

12. The subroutine *PSSR is called for I/O operation errors that are not handled by the subroutine in section 6. This subroutine checks to see whether the program device is already acquired when an acquire
operation is requested and if so, the second acquire
is ignored. Otherwise, the program ends.

SEQUENCE IND DO LAST PAGE PROGRAM
NUMBER *.....2....*....3....*....4....*....5....*....6....*....7....* USE NUM UPDATE LINE ID

Source Listing

100 //*****************************************************************************
200 // THIS PROGRAM WILL HANDLE THE REQUEST FOR EITHER A BUYER
300 // NUMBER OR AN ITEM NUMBER. THIS IS ACCOMPLISHED BY MAKING
400 // THE DATA BASE FILE STRUCTURE (KEY LENGTH, KEY POSITION,
500 // RECORD LENGTH, RECORD SIZE, ETC.) THE SAME FOR BOTH FILES
600 // WITH ONLY THE RECORD CONTENTS DIFFERENT.
700 // THIS PROGRAM ENDS WHEN A DETACH REQUEST IS RECEIVED FROM
800 // THE SOURCE PROGRAM.
1000 //*****************************************************************************
1100 //
1200 //*****************************************************************************
1300 // FCFILE CF E WORKSTN
1400 //*****************************************************************************

Figure D-19 (Part 1 of 5). Target Program Example—RTDINT
SEQUENCE IND DO LAST PAGE PROGRAM

C READ TAG

C READ RCVPART 9950 2 3

C MAJCOD CABG'T:03' ERROR

C +IN44 CABELQ'1' END DET RCV? END

C MOVE RECNO2 DSEQ

C MAJMIN CABEQ'0000' XMIT RCYTNRND?

C +IN40 CABELQ'1' XMIT RCYTNRND?

C GOTO READ NO,TRY AGAIN

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************

C*************************************************************************
SEQUENCE IND DO LAST PAGE PROGRAM

NUMBER /c5197...1....+....2....+....3....+....4....+....5....+....6....+....7.../c5197 USE NUM UPDATE LINE ID
7800 C 97 MOVE '1' +INZO SEND FAIL 11/21/88
7900 C WRITEMODE DATA W/DTR 10/13/87
8000 C MAJCOX CABS'03' ERROR 10/13/87
8100 C MOVE '0' +INZO RESET IND 11/21/88
8200 C GOTO READ 10/13/87
8300 C 10/13/87
8400 C******************************************************************************
8500 C* 03/17/89
8600 C* IF ANY IFC FILE ERROR OCCURS, PRINT THE ERROR MESSAGE, AND 03/17/89
8700 C* THEN END THE JOB. 03/17/89
8800 C* 03/17/89
8900 C******************************************************************************
9000 C 03/17/89
9100 C ERROR 03/17/89
9200 C EXCPMMER 03/17/89
9300 C END 03/17/89
9400 C 03/17/89
9500 C SETON LR 03/17/89
9600 C 03/17/89
9700 C******************************************************************************
9800 C* 03/17/89
9900 C* THIS IS THE PROGRAM EXCEPTION/ERROR SUBROUTINE THAT 03/17/89
10000 C* RECEIVES CONTROL WHEN AN EXCEPTION OR ERROR OCCURS 03/17/89
10100 C* AFTER AN 1/O IS ISSUED TO AN ICF PROGRAM DEVICE AND 03/17/89
10200 C* THERE IS A NON-ZERO VALUE UPDATED IN THE RPG STATUS 03/17/89
10300 C* FIELD (ERR). THIS ROUTINE CHECKS FOR STATUS VALUES THAT 03/17/89
10400 C* RELATE TO ICF OPERATION. 03/17/89
10500 C* IF THE PROGRAM DEVICE IS ALREADY ACQUIRED, THE EXCEPTION 03/17/89
10600 C* IS IGNORED, OTHERWISE THE PROGRAM IS TERMINATED. 03/17/89
10700 C* 03/17/89
10800 C******************************************************************************
10900 C******************************************************************************
11000 C *PSR BEGSR 10/14/87
11100 C MOVE ' ' RETURN 6 DEFAULT 10/14/87
11200 C ERR CABEQ/1285 ENDPSR ALREADY ACQ? 10/14/87
11300 C MOVE '/CANCL' RETURN JOB ENDS * 10/14/87
11400 C ENDPSR ENDSRRETURN BACK TO MAIN 10/14/87
11500 C******************************************************************************
11600 OQPRINT E 1 MMERR 10/13/87
11700 O 21 'ERROR ON IFC FILE' 10/05/90
11800 O 34 'MAJOR/MINOR:' 10/13/87
11900 O MAJCOX 37 10/13/87
12000 O 38 '/' 10/13/87
12100 O MINCOX 40 10/13/87
12200 O 49 'FORMAT:' 10/13/87
12300 O FMTNM 60 10/13/87
12400 O 69 'PGMDEV:' 10/13/87
12500 O PGMDVY 80 10/13/87
* 6103 12501 OVERFLOW INDICATOR OA ASSIGNED TO FILE QPRINT.
D000000 OUTPUT FIELDS FOR RECORD SNDPART FILE CFILE FORMAT SNDPART.
D000001 RECYP 1 CHAR 1
D000002 ITIMNO 7 CHAR 6
Figure D-19 (Part 3 of 5). Target Program Example —RTDINT

Intrasystem Communications Programming V4R1
Cross Reference

File and Record References:

01 CFILE WORKSTN 12000
   RCVPART 12000 B000000 4100
   SNDPART 12000 A000000 7900 D000000
02 PFILE DISK 15000 6200
   DBREC 15000 C000000
03 OPRINT PRINTER 16000 11600 12501

Field References:

1285 END TAG 4/77/82 8200
1285 ENDPSR ENDSR 112/114/116
1285 ERR Z(5) 19/112/116
1285 FEEDBK DS(4) 12/17/125
1285 FILL1 A(13) B000000 D000000
1285 FILL4 A(64) B/3/12
1285 FMTNM A(1) 2/123/125
1285 ITEMNO A(6) A000000 D000000
1285 Item References:

1285 LR 95/76/78
1285 OA 16/125/126
1285 01285 LITERAL 11200
1285 03' LITERAL 4200 8000
1285 'I' LITERAL 4300 4600 7600 7800
1285 '00000' LITERAL 7500
1285 '03' LITERAL 4200 8000
1285 'I' LITERAL 4300 4600 7600 7800
1285 '01285' LITERAL 11200

Indicator References:

1285 LR 95/76/78
1285 OA 16/125/126

Figure D-19 (Part 4 of 5). Target Program Example — RTDINT
Message Summary

* QRG6103 Severity: 00 Number: 1
  Message . . . : No Overflow Indicator is specified but an 
  indicator is assigned to a file and automatic skip to 6 is 
  generated.

* QRG7031 Severity: 00 Number: 11
  Message . . . : The Name or indicator is not referenced.

* QRG7089 Severity: 00 Number: 1
  Message . . . : The RPG provides Separate-Indicator area for 
  file.

* * * * * E N D O F M E S S A G E S U M M A R Y * * * * *

Final Summary

Message Count: (by Severity Number)

TOTAL 00 10 20 30 40 50
13 13 0 0 0 0

Program Source Totals:

Records . . . . : 125
Specifications . . . . : 53
Table Records . . . . : 0
Comments . . . . : 72

PRM has been called.

Program RTDINT is placed in library INTLIB. 00 highest Error-Severity-Code.

* * * * * E N D O F C O M P I L A T I O N * * * * *

Figure D-19 (Part 5 of 5). Target Program Example —RTDINT
Bibliography

The publications listed here provide additional information about topics described or referred to in this book. These books are listed with their full title and order number.

**AS/400 Books**
The following AS/400 books contain additional information you may need when developing application programs that use intrasystem communications support.

- **ICF Programming**, SC41-5442, provides the application programmer with information needed to write programs that use AS/400 communications and the OS/400 intersystem communications function (OS/400-ICF).
- **Communications Configuration**, SC41-5401, contains general configuration information, including detailed descriptions of network interface, line, controller, device, mode, and class-of-service descriptions, configuration lists and connection lists.
- **DDS Reference**, SC41-5712, contains information about coding data description specifications for files.
- **C/400* User's Guide**, SC09-1347, provides the information needed to write, test, and maintain C/400 programs for the AS/400 system.
- **COBOL/400 User's Guide**, SC09-1812, provides the information needed to write, test, and maintain COBOL/400 programs for the AS/400 system.
- **RPG/400 User's Guide**, SC09-1816, provides the information needed to write, test, and maintain RPG/400 programs for the AS/400 system.
- **CL Reference**, SC41-5722, contains the commands, command parameters, and syntax for the commands used in this book.
- **Work Management**, SC41-5306, contains information about how to create an initial work management environment and how to change it.
- **System Operation**, SC41-4203, provides information on how to use the system unit operator display.

**System/36 Communications Books**
The following book provides a description of the Intra Subsystem on the System/36, and information about setting up and configuring the Intra Subsystem, communications operations, and return codes:


The following two books provide information and examples about the interactive communications feature, a feature of the System Support Program Product on the System/36 that allows a program to communicate interactively with another program or system:

- **Interactive Communications Feature: Base Subsystems Reference**, SC21-9530.
- **Interactive Communications Feature: Guide and Examples**, SC21-7911.

The following book provides an overview for programming in the System/36 environment:

- **System/36 Environment Programming**, SC41-4730.
Index

A
acquire operation 4-3
Add ICF Device Entry (ADDICFDEVE) command 4-1
ADDICFDEVE command 4-1
advanced program-to-program communications (APPC)
definition C-1
allow-write function
definition 4-6
using C-3
APPC (advanced program-to-program communications) C-1
application
communications 1-2
close 5-2
cancel 4-4
cancel-invite 4-6
general 5-1
input 5-1
open/acquire 5-1
programs 4-1
testing 1-2, C-1
AS/400 manuals H-1
ASCVRYOFF parameter 3-1
asynchronous communications
definition C-2
AUT parameter 2-1
authority (AUT) parameter 2-1

B
BATCH parameter 4-2
bibliography H-1
binary synchronous communications (BSC)
definition C-2
binary synchronous communications equivalence link (BSCEL)
definition C-2
BSC (binary synchronous communications)
definition C-2
BSCEL (binary synchronous communications equivalence link)
definition C-2

C
C Set ++ programming language
cancel function
definition 4-5
cancel-invite function
definition 4-6
CFGOBJ parameter 3-1
CFGTYPE parameter 3-1
Change Device Description (Intrasystem) (CHGDEVINTR)command 2-1
Change ICF Device Entry (CHGICFDEVE) command 4-1
Change ICF File (CHGICFF) command 4-1
CHGDEVINTR command 2-1
CHGICFDEVE command 4-1
CHGICFF command 4-1
close operation
cancel-invite considerations 5-2
definition 4-7
CMNTYPE parameter 4-2
COBOL/400 programming language
source program D-14	target program D-33
command prompt 2-1
commands
Add ICF Device Entry (ADDICFDEVE) 4-1
Change Device Description (Intrasystem) (CHGDEVINTR) 2-1
Change ICF Device Entry (CHGICFDEVE) 4-1
Change ICF File (CHGICFF) 4-1
command prompt 2-1
Create Device Description (Intrasystem) (CRTDEVINTR) 2-1
Create ICF File (CRTICFF) 4-1
Delete File (DLTF) 4-1
direct entry 2-1
Display Field Description (DSPFFD) 4-1
Display File Description (DSPFD) 4-1
testing 2-1
Override ICF Device Entry (OVRICFDEVE) 4-1
Override ICF File (OVRICFF) 4-1
Remove ICF Device Entry (RMVICFDEVE) 4-1
Vary Configuration (VRYCFG) 3-1
communications
application testing 1-2
intrasystem
configuration 2-1
cancel-invite considerations 5-1
operations 4-3
communications type (CMNTYPE) parameter 4-2
configuring intrasystem communications 2-1
confirm function
advanced program-to-program communications (APPC) C-1
cancel-invite considerations 5-2
definition 4-4
finance communications C-3
retail communications C-4
sending data 4-4
considerations
    applications 5-1
    close operation 5-2
    confirm function 5-2
    general 5-1
    input 5-1
    intrasystem communications 5-1
    open/acquire 5-1
    performance 5-2

conversation types C-1
Create Device Description (Intrasystem) (CRTDEVINTR) command 2-1
Create ICF File (CRTICFF) command 4-1
CRTDEVINTR command 2-1
CRTICFF command 4-1

D

data
    management 1-1
    queue 4-5
    receiving 4-4, C-3, C-4
    sending 4-4, C-4
    sense C-4, C-5

data description specifications (DDS) keywords A-2

DDS keywords A-2
Delete File (DLTF) command 4-1
detach function
    asynchronous communications C-2
    binary synchronous communications equivalence link (BSCEL) C-2
    definition 4-6
    ending transactions 4-6
    retail communications C-4

DEVD parameter 2-1
device description (DEVD) parameter 2-1
direct entry of commands 2-1
Display Field Description (DSPFFD) command 4-1
display file 4-5
Display File Description (DSPFD) command 4-1
DLTF command 4-1
DSPFD command 4-1
DSPFFD command 4-1

e
end-of-group function
    binary synchronous communications equivalence link (BSCEL) C-2
    definition 4-4
    finance communications C-3
    retail communications C-4
    sending data 4-4

Systems Network Architecture Uplink Facility (SNUF) C-5
end-of-session considerations 5-2

end-of-session function
definition 4-7

ending
    sessions 4-7
    transactions 4-6

every of commands
    command prompt 2-1
direct 2-1
evoke function
    advanced program-to-program communications (APPC) C-1
    asynchronous communications C-2
    binary synchronous communications equivalence link (BSCEL) C-2
    definition 4-3
    retail communications C-4
    starting transactions 4-3

Systems Network Architecture Uplink Facility (SNUF) C-5
examples
    COBOL/400 source program D-14
    COBOL/400 target program D-33

commands
    Create Device Description (Intrasystem) (CRTDEVINTR) command 2-1
    Vary Configuration (VRYCFG) 3-1
    VRYCFG 3-1

device description 2-1
ILE C/400 source program D-1
ILE C/400 target program D-9
intrasystem communications configuration 2-1
performance consideration 5-1
program D-1
return code usage 4-8, 4-9
RPG/400 source program D-39
RPG/400 target program D-56
single-session inquiry program D-1
two-session inquiry program D-14

F

fail function
    advanced program-to-program communications (APPC) C-1
    asynchronous communications C-2
    binary synchronous communications equivalence link (BSCEL) C-3
    definition 4-5
    problem notification 4-5

Systems Network Architecture Uplink Facility (SNUF) C-5
failed program start requests B-24

feedback area 4-8
    return code usage
    error condition 4-9

X-2 Intrasystem Communications Programming V4R1
feedback area (continued)
  using
    error condition 4-9

file commands
  Add ICF Device Entry (ADDICFDEVE) 4-1
  Change ICF Device Entry (CHGICFDEVE) 4-1
  Change ICF File (CHGICFF) 4-1
  Create ICF File (CRTICFF) 4-1
  Delete File (DLTF) 4-1
  Display Field Descriptions (DSPFD) 4-1
  Display File Descriptions (DSPFFD) 4-1
  Override ICF Device Entry (OVRICFDEVE) 4-1
  Override ICF File (OVRICFF) 4-1
  Remove ICF Device Entry (RMVICFDEVE) 4-1

FILE parameter 4-2

finance communications
  definition C-3

FMTSLT parameter 4-2

force-data function
  advanced program-to-program communications (APPC) C-1
  definition 4-4
  finance communications C-3
  retail communications C-4
  sending data 4-4

format-name function 4-4
  definition 4-4

function-management-header data C-5

function-management-header function
  definition 4-4
  using C-2

functions
  allow-write 4-6, C-3
  cancel 4-5
  cancel-invite 4-6
  detach
    asynchronous communications C-2
    binary synchronous communications equivalence link (BSCEL) C-2
    ending transactions 4-6
    retail communications C-4
  end-of-group
    binary synchronous communications equivalence link (BSCEL) C-2
    finance communications C-3
    retail communications C-4
    sending data 4-4
    Systems Network Architecture Uplink Facility (SNUF) C-5
  end-of-session 4-7
  evoke
    advanced program-to-program communications (APPC) C-1
    asynchronous communications C-2
    binary synchronous communications equivalence link (BSCEL) C-2
    retail communications C-4

fail
  advanced program-to-program communications (APPC) C-1
  asynchronous communications C-2
  binary synchronous communications equivalence link (BSCEL) C-3
  problem notification 4-5
  Systems Network Architecture Uplink Facility (SNUF) C-5

force-data
  advanced program-to-program communications (APPC) C-1
  finance communications C-3
  retail communications C-4
  sending data 4-4

format-name 4-4

function-management-header 4-4, C-2

invite
  finance communications C-3
  receiving data 4-4
  retail communications C-4

keyword A-2

negative-response 4-5

request-to-write 4-6, C-3

respond-to-confirm 4-6

subdevice selection 4-4

timer 4-6

G

general considerations of intrasystem communications applications 5-1

get-attributes operation
  definition 4-6

I

ICF (intersystem communications function)

ILE C/400 programming language
  functions A-1
  source program D-1
  target program D-9

ILE COBOL/400 programming language
  procedure statements A-1

ILE RPG/400 programming language
  operation codes A-1

indicators
  receive-cancel 4-8
  receive-confirm 4-7
  receive-detach 4-8
  receive-end-of-group 4-7
indicators (continued)
  receive-fail 4-8
  receive-function-management-header 4-8
  receive-negative-response 4-8
  receive-turnaround 4-8
input considerations 5-1
input/output feedback area 4-8
intersystem communications function (ICF)
  data management 1-1
  definition 1-1
  file 4-1, 4-5
  file commands
    Add ICF Device Entry (ADDICFDEVE) 4-1
    Change ICF Device Entry (CHGICFDEVE) 4-1
    Change ICF File (CHGICFF) 4-1
    Create ICF File (CRTICFF) 4-1
    Delete File (DLTF) 4-1
    Display Field Description (DSPFFD) 4-1
    Display File Description (DSPFD) 4-1
    Override ICF Device Entry (OVRICFDEVE) 4-1
    Override ICF File (OVRICFF) 4-1
    Remove ICF Device Entry (RMVICFDEVE) 4-1
language operations A-1
intrasystem
  application programs 4-1
  communications
    application considerations 5-1
    configuration 2-1
    definition 1-1
    device description 2-1
    overview 1-1
    performance considerations 5-2
    support 1-1, 3-1
  testing communications applications 1-2
invite function
  definition 4-4
  finance communications C-3
  receiving data 4-4
  retail communications C-4

J
  jobs 5-2

K
  keyword functions A-2
  keywords A-2

L
  language operations A-1

M
  manuals
    AS/400 H-1

N
  name of device description (CFGOBJ) parameter 3-1
  negative-response function
    definition 4-5
  number of sessions
    asynchronous communications C-2
    binary synchronous communications equivalence link (BSCEL) C-3
    finance communications C-4
    retail communications C-5
    Systems Network Architecture Uplink Facility (SNUF) C-5

O
  online messages C-3
  ONLINE parameter 2-1
  open operation 4-3
  open/acquire
    considerations 4-3
    operation 4-3
  operations
    acquire 4-3
    close 4-7
    communications 4-3
    get-attributes 4-6
    open 4-3
    open/acquire 4-3
    output C-1
      read
        asynchronous communications C-2
        finance communications C-4
        receiving data 4-4
        retail communications C-5
        read-from-invited-program-devices 4-5
        release 4-7
      write
        asynchronous communications C-2
        finance communications C-3
        retail communications C-4
        sending data 4-4
  output operations C-1
  Override ICF Device Entry (OVRICFDEVE) command 4-1
  Override ICF File (OVRICFF) command 4-1
  overview of intrasystem communications 1-1
  OVRICFF command 4-1

P
  parameters
    ADDICFDEVE command
      BATCH 4-2
parameters (continued)
ADDICFDEVE command (continued)
  CMNTYPE 4-2
  FILE 4-2
  FMTSLT 4-2
  PGMDEV 4-2
  RMTLOCNAME 4-2
ASCVRYOFF 3-1
authority (AUT) 2-1
BATCH 4-2
CHGDEVINTR command
  AUT 2-1
  DEVD 2-1
  ONLINE 2-1
  TEXT 2-1
CHGICFDEVE command
  BATCH 4-2
  CMNTYPE 4-2
  FILE 4-2
  FMTSLT 4-2
  PGMDEV 4-2
  RMTLOCNAME 4-2
communications type (CMNTYPE) 4-2
CRTDEVINTR command
  AUT 2-1
  DEVD 2-1
  ONLINE 2-1
  RMTLOCNAME 2-1
  TEXT 2-1
device description (DEVD) 2-1
FILE 4-2
name of device description (CFGOBJ) 3-1
ONLINE 2-1
OVRICFDEVE command
  BATCH 4-2
  CMNTYPE 4-2
  FILE 4-2
  FMTSLT 4-2
  PGMDEV 4-2
  RMTLOCNAME 4-2
  SECURE 4-2
program device name (PGMDEV) 4-2
RANGE 3-1
record format selection (FMTSLT) 4-2
remote location name (RMTLOCNAME) 2-1, 4-2
SECURE 4-2
STATUS 3-1
TEXT 2-1
type of configuration description (CFGTYPE) 3-1
VRYCFG command
  ASCVRYOFF 3-1
  CFGOBJ 3-1
  CFGTYPE 3-1
  RANGE 3-1
  STATUS 3-1
  VRYWAIT 3-1
VRYWAIT 3-1

performance considerations 5-2
PGMDEV parameter 4-2
PIP (program initialization procedure)
definition 4-3
prestarting jobs for program start requests 5-2
problem notification 4-5
program
device entry commands 4-2
examples D-1
start requests 5-2, C-3
testing C-1
program device name (PGMDEV) parameter 4-2
program initialization procedure (PIP)
definition 4-3

R
RANGE parameter 3-1
read
function
  finance communications C-3
  retail communications C-4
operation
  asynchronous communications C-2
  finance communications C-4
  receiving data 4-4
  retail communications C-5
read operation function
definition 4-4
read-from-invited-program-devices operation
definition 4-5
receive-cancel response indicator
definition 4-8
receive-confirm response indicator
definition 4-7
receive-detach response indicator
definition 4-8
receive-end-of-group response indicator
definition 4-7
receive-fail response indicator
definition 4-8
receive-function-management-header response indicator
definition 4-8
receive-negative-response response indicator
definition 4-8
receive-turnaround indication C-3
receive-turnaround response indicator
definition 4-8
receiving data 4-4, C-3, C-4
record
  blocking C-3
  length C-1, C-3
record format selection (FMTSLT) parameter 4-2
related printed information H-1
release function
  considerations 5-2
release operation
   definition 4-7
remote location name (RMTLOCNAME) parameter 2-1, 4-2
Remove ICF Device Entry (RMVICFDEV) command 4-1
request-to-write function
   definition 4-6
   using C-3
respond-to-confirm function
   definition 4-6
response indicator
   definition 4-7
response indicators
   receive-cancel 4-8
   receive-confirm 4-7
   receive-detach 4-8
   receive-end-of-group 4-7
   receive-fail 4-8
   receive-function-management-header 4-8
   receive-negative-response 4-8
   receive-turnaround 4-8
   using 4-7
retail communications
   definition C-4
return codes
   detailed descriptions of B-1
   using 4-8, 4-9
RMTLOCNAME parameter 2-1, 4-2
RMVICFDEV command 4-1
RPG/400 programming language
   source program D-39
   target program D-56

S
SECURE parameter 4-2
sending data 4-4, C-4
sense data
   definition 4-6
   finance communications C-4
   retail communications C-5
sessions
   ending 4-7
   starting 4-3
single-session inquiry program
   ILE C/400 source program example D-1
   ILE C/400 target program example D-9
SNUF (Systems Network Architecture Uplink Facility)
   definition C-5
source program
   COBOL/400 two-session inquiry example D-14
   ILE C/400 single-session inquiry example D-1
   RPG/400 two-session inquiry example D-39
starting
   sessions 4-3
   transactions 4-3
STATUS parameter 3-1
subdevice selection function
   definition 4-4
system messages C-5
system-supplied formats A-2
System/36 manuals H-1
Systems Network Architecture Uplink Facility (SNUF)
   definition C-5
T
target program
   COBOL/400 two-session inquiry example D-33
   ILE C/400 single-session inquiry example D-9
   RPG/400 two-session inquiry example D-56
testing application programs
   advanced program-to-program communications (APPC) C-1
   asynchronous communications C-2
   binary synchronous communications (BSC) C-2
   binary synchronous communications equivalence link (BSCEL) C-2
   communications applications 1-2
   finance communications C-3
   retail communications C-4
   using intrasystem communications C-1
TEXT parameter 2-1
timer function
   definition 4-6
transactions
   definition 4-3
   ending 4-6
   starting 4-3
translation C-2
two-session inquiry program
   COBOL/400 source program example D-14
   COBOL/400 target program example D-33
   RPG/400 source program example D-39
   RPG/400 target program example D-56
type of configuration description (CFGTYPE) parameter 3-1
V
variable buffer management (VARBUFMGT) C-1
Vary Configuration (VRYCFG) command 3-1
vary off 3-1
vary on 3-1
VRYCFG command 3-1
VRYWAIT parameter 3-1
W
what you should know vii
who should use this book vii
write operation
  asynchronous communications  C-2
  definition  4-4
  finance communications  C-3
  retail communications  C-4
  sending data  4-4
writing application programs  4-1
Reader Comments—We'd Like to Hear from You!

AS/400
Intrasystem Communications
Programming
Version 4
Publication No. SC41-5447-00

Overall, how would you rate this manual?

<table>
<thead>
<tr>
<th>Overall satisfaction</th>
<th>Very Satisfied</th>
<th>Satisfied</th>
<th>Dissatisfied</th>
<th>Very Dissatisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How satisfied are you that the information in this manual is:

- Accurate
- Complete
- Easy to find
- Easy to understand
- Well organized
- Applicable to your tasks

THANK YOU!

Please tell us how we can improve this manual:

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

May we contact you to discuss your responses? ___ Yes ___ No
Phone: (____) ___________ Fax: (____) ___________ Internet: ___________

To return this form:
- Mail it
- Fax it
  United States and Canada: 800+937-3430
  Other countries: (+1)+507+253-5192
- Hand it to your IBM representative.

Note that IBM may use or distribute the responses to this form without obligation.

Name

Address

Company or Organization

Phone No.