Finance Communications Programming

Version 4
<table>
<thead>
<tr>
<th>Chapter 1. Introduction to Finance Communications</th>
<th>1-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance Communications Environment</td>
<td>1-1</td>
</tr>
<tr>
<td>Finance Communications Network Example</td>
<td>1-2</td>
</tr>
<tr>
<td>Finance Communications Using SNA Pass-Through Support</td>
<td>1-2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2. Finance Communications Support</th>
<th>2-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance Functions</td>
<td>2-1</td>
</tr>
<tr>
<td>Concurrent Sessions</td>
<td>2-2</td>
</tr>
<tr>
<td>Security</td>
<td>2-2</td>
</tr>
<tr>
<td>Error Handling and Recovery</td>
<td>2-2</td>
</tr>
<tr>
<td>System Monitor Session</td>
<td>2-3</td>
</tr>
<tr>
<td>Finance Interfaces</td>
<td>2-3</td>
</tr>
<tr>
<td>Intersystem Communications Function Finance Support</td>
<td>2-3</td>
</tr>
<tr>
<td>Non-Intersystem Communications Function Finance Support</td>
<td>2-3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 3. Configuring Finance Support</th>
<th>3-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring the AS/400 System for Finance Communications</td>
<td>3-1</td>
</tr>
<tr>
<td>Configuring the Line Description</td>
<td>3-1</td>
</tr>
<tr>
<td>Configuring the Controller Description</td>
<td>3-1</td>
</tr>
<tr>
<td>Configuring the Device Description</td>
<td>3-2</td>
</tr>
<tr>
<td>Using 3270 Devices</td>
<td>3-3</td>
</tr>
<tr>
<td>Configuring the AS/400 System for SNA Pass-Through Communications</td>
<td>3-4</td>
</tr>
<tr>
<td>Associating Finance and SNA Pass-Through Devices</td>
<td>3-5</td>
</tr>
<tr>
<td>Using the Work with Table Commands to Configure Non-Intersystem Communications Function (Non-ICF) Finance</td>
<td>3-5</td>
</tr>
<tr>
<td>Work with Device Table Command</td>
<td>3-5</td>
</tr>
<tr>
<td>Work with User Table Command</td>
<td>3-6</td>
</tr>
<tr>
<td>Work with Program Table Command</td>
<td>3-7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4. Running Finance Support</th>
<th>4-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Varying Finance On and Off</td>
<td>4-1</td>
</tr>
<tr>
<td>Using the Vary On Configuration Command</td>
<td>4-1</td>
</tr>
<tr>
<td>Using the Work with Configuration Status Command</td>
<td>4-2</td>
</tr>
<tr>
<td>Defining Communications Entries</td>
<td>4-2</td>
</tr>
<tr>
<td>Starting the Finance Subsystem</td>
<td>4-2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5. Writing Intersystem Communications Function Finance Application Programs</th>
<th>5-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Intersystem Communications Function File Commands</td>
<td>5-1</td>
</tr>
<tr>
<td>Starting a Session</td>
<td>5-2</td>
</tr>
<tr>
<td>Starting a Session with a 47xx or Financial Branch System Services Controller</td>
<td>5-3</td>
</tr>
<tr>
<td>Starting a Session with a 3694 Document Processor</td>
<td>5-4</td>
</tr>
<tr>
<td>Open/Acquire Operation</td>
<td>5-4</td>
</tr>
<tr>
<td>Sending Data</td>
<td>5-4</td>
</tr>
<tr>
<td>Write Operation</td>
<td>5-5</td>
</tr>
<tr>
<td>Force-Data Function</td>
<td>5-5</td>
</tr>
<tr>
<td>End-Of-Group Function</td>
<td>5-5</td>
</tr>
<tr>
<td>Function-Management-Header Function</td>
<td>5-5</td>
</tr>
<tr>
<td>Receiving Data</td>
<td>5-5</td>
</tr>
<tr>
<td>Read Operation</td>
<td>5-5</td>
</tr>
<tr>
<td>Invite Function</td>
<td>5-6</td>
</tr>
<tr>
<td>Read-from-Invited-Program-Devices Operation</td>
<td>5-6</td>
</tr>
<tr>
<td>Waiting for a Display File, an ICF File, and a Data Queue</td>
<td>5-6</td>
</tr>
<tr>
<td>Notifying the Remote Program of Problems</td>
<td>5-7</td>
</tr>
<tr>
<td>Fail Function</td>
<td>5-7</td>
</tr>
<tr>
<td>Cancel Function</td>
<td>5-8</td>
</tr>
<tr>
<td>Negative-Response Function</td>
<td>5-8</td>
</tr>
<tr>
<td>Using Additional Functions and Operations</td>
<td>5-8</td>
</tr>
<tr>
<td>Cancel-Invite Function</td>
<td>5-8</td>
</tr>
<tr>
<td>Timer Function</td>
<td>5-8</td>
</tr>
<tr>
<td>Get-Attributes Operation</td>
<td>5-8</td>
</tr>
<tr>
<td>Ending a Session</td>
<td>5-8</td>
</tr>
<tr>
<td>Release Operation</td>
<td>5-9</td>
</tr>
<tr>
<td>End-of-Session Function</td>
<td>5-9</td>
</tr>
<tr>
<td>Close Operation</td>
<td>5-9</td>
</tr>
<tr>
<td>Using Response Indicators</td>
<td>5-9</td>
</tr>
<tr>
<td>Receive End-of-Group</td>
<td>5-10</td>
</tr>
<tr>
<td>Receive Function-Management-Header</td>
<td>5-10</td>
</tr>
<tr>
<td>Receive Negative-Response</td>
<td>5-10</td>
</tr>
<tr>
<td>Using Input/Output Feedback Area</td>
<td>5-10</td>
</tr>
<tr>
<td>Using Return Codes</td>
<td>5-10</td>
</tr>
</tbody>
</table>
Chapter 6. Writing Non-Intersystem Communications Function Application

Programs .................................. 6-1
Using the Submit Finance Job Command 6-1
Parameters .................................. 6-2
Error Handling .............................. 6-3
Supervising Finance Jobs ................... 6-3
Data Flow Examples .......................... 6-4
Processing Transactions ...................... 6-6
Using Finance Input/Output Manager ........ 6-7
Finance Input/Output Manager Routines 6-8
Error Handling .............................. 6-10
Using Finance Input/Output Manager with the Submit Finance Job Command .......... 6-11
Using Finance Input/Output Manager without the Submit Finance Job Command .......... 6-13
Using User-Defined Data Stream .......... 6-14
Formats ..................................... 6-15
Data Streams ................................ 6-16
3694 Communications with User-Defined Data Stream .......................... 6-19
Example of User-Defined Data Stream .......... 6-21

Chapter 7. Finance Considerations ....... 7-1
Intersystem Communications Function .... 7-1
Programming Applications .................. 7-1
Performance ................................ 7-1
Non-Intersystem Communications Function .... 7-2
Programming Applications .................. 7-2
Security ..................................... 7-6
Controller Applications ..................... 7-6
Systems Network Architecture (SNA) ......... 7-7
INIT-SELF Command Field Format .......... 7-7
BIND Command Field Format ............... 7-8

Appendix A. Language Operations, DDS Keywords, and System-Supplied Formats A-1
Using Language Operations ................ A-1
Intersystem Communications Function Operations ........ A-1
Intersystem Communications Function Language Statements ................ A-2
Data Description Specifications Keywords .......... A-3
System-Supplied Formats .......... A-3

Appendix B. Return Codes, Messages, and Sense Codes .... B-1
Return Codes ................................ B-1
Major Code 00 ................................ B-1
Major Code 02 ................................ B-2
Major Code 03 ................................ B-3
Major Code 04 ................................ B-4
Major Codes 08 and 11 ...................... B-5

Appendix C. Mapping Intersystem Communications Function Operations to Systems Network Architecture Commands .......... C-1

Appendix D. 4701 Finance Controller Diskette Download ........ D-1
Using the Send Finance Diskette Image Command ........ D-1

Appendix E. Intersystem Communications Function Finance Example Programs .... E-1
COBOL/400 Source Program for Local System ................ E-1
Program Files .......... E-1
RPG/400 Source Program for Local System—Prestarted Job Example .... E-27
Program Files .......... E-27
ILE C/400 Source Program for Local System—Prestarted Job Example .... E-27
Program Files .......... E-27

Appendix F. Non-Intersystem Communications Function Finance Example Programs .... F-1
COBOL/400 and RPG/400 Source Programs for the Local System ........ F-1
Program File .......... F-1
COBOL/400 Program Explanation .......... F-3
RPG/400 Program Explanation .......... F-18

Appendix G. Configuration Examples .......... G-1
Configuration Example for FBSS Controller Using SDLC .......... G-1
Program Explanation .......... G-1
Configuration Example for FBSS Controller Using Token-Ring Network .......... G-2
Program Explanation .......... G-2
Configuration Example for FBSS Controller Using X.25 .......... G-3
Program Explanation .......... G-4
Configuration Example for 4702 Controller Using SDLC .......... G-4
Program Explanation .......... G-6
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibliography</td>
<td>H-1</td>
</tr>
<tr>
<td>AS/400 Books</td>
<td>H-1</td>
</tr>
<tr>
<td>Programming Language Books</td>
<td>H-1</td>
</tr>
<tr>
<td>Personal Banking Machine Books</td>
<td>H-1</td>
</tr>
<tr>
<td>Financial Branch System Services (FBSS)</td>
<td>H-2</td>
</tr>
<tr>
<td>Systems Network Architecture (SNA) Books</td>
<td>H-2</td>
</tr>
<tr>
<td>Miscellaneous Books</td>
<td>H-2</td>
</tr>
<tr>
<td>Index</td>
<td>X-1</td>
</tr>
<tr>
<td>G-5. FBSS Communication Servers Display</td>
<td>G-11. FBSS X.25/SNA Configuration Display</td>
</tr>
<tr>
<td>G-6. FBSS SSCP Names Display</td>
<td>G-12. FBSS X.25/SNA Permanent Circuit Configuration Display</td>
</tr>
<tr>
<td>G-7. FBSS Token-Ring Communications Display</td>
<td>G-13. FBSS Session-Id and LU Assignments Display</td>
</tr>
<tr>
<td>G-8. FBSS Session-Id and LU Assignments Display</td>
<td>G-14. AS/400 Configurations Example for Finance Communications</td>
</tr>
<tr>
<td>G-9. FBSS Communication Servers Display</td>
<td>G-15. CPGEN Program</td>
</tr>
</tbody>
</table>
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 to use the finance communications support. This book documents General-Use Programming Interface and Associated Guidance Information provided by OS/400 program.

General-Use programming interfaces allow the customer to write programs that obtain the services of OS/400 program.

The information in Chapter 6, “Writing Non-Intersystem Communications Function Application Programs” is intended to help the customer write finance application programs using the non-ICF interface. It contains general-use programming interfaces, which allow the customer to write programs that use the services of the OS/400 program.

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:

| AD/Cycle | Application System/400 |
| APPN | AS/400 |
| C/400 | COBOL/400 |
| C Set ++ for OS/400 | IBM |
| ILE | |
| Operating System/400 | OS/400 |
| Personal System/2 | PS/2 |
| RPG/400 | SAA |
| System/370 | System/390 |
| Systems Application Architecture | 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.
About Finance Communications Programming (SC41-5449)

This book contains the commands and procedures for setting up and using both the intersystem communications function (ICF) finance support and the non-ICF finance support for the AS/400 system.

Throughout this book, the terms “controller” and “processor” are used interchangeably depending on the machine type.

For a list of publications related to this book, see the “Bibliography.”

Who Should Use This Book

This book is intended for system administrators, system operators, and application programmers.

For more information on basic communications you can also refer to the Discover/Education* course in the communications module. The Discover/Education course is separately orderable.

Before you use the material in this book, you should be familiar with the book, CL Programming. This book contains a wide-ranging discussion of AS/400 system topics, including a general discussion of objects and libraries and control language (CL) programming, as well as a discussion on controlling flow and communications. To write your transaction processing applications, you must know the programming language of the system and how to enter and create a program at an AS/400 work station.

Depending on the interface you use to communicate between the AS/400 system and the attached finance devices, you need to understand varying amounts of the information found in the Communications Configuration book, and the ICF Programming book.

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 Finance Communications

This chapter is an introduction to AS/400* finance communications. It also includes an overview of the intersystem communications function (ICF) and non-ICF communications support information, and an example of a finance communications network.

Finance Communications Environment

Finance communications uses high-level language operations and communications functions that allow you to communicate between an AS/400 system and finance controllers, or between a System/370* or System/390* host system and finance controllers attached to an AS/400 system, providing a banking environment communications system. Finance communications allows programs in the supported high-level languages C Set ++ for OS/400*, (Integrated Language Environment (ILE) C/400*, ILE COBOL/400*, and ILE RPG/400* languages) on an AS/400 system to communicate with IBM* 3694, 4701, 4702, 4730, 4731, 4732, and 4736 finance processors and personal computers or Personal System/2* work stations using Financial Branch System Services (FBSS) software.

Note: The FBSS software must be version 2.2 or later when using DOS.

AS/400 finance communications includes two separate methods of communication: ICF finance communications and non-ICF finance communications. Both methods communicate using the Systems Network Architecture logical unit (SNA LU) type 0 (LU0) primary protocol.

Figure 1-1 provides an overview of both the ICF and non-ICF finance communications methods.

---

**Figure 1-1. Overview of ICF and Non-ICF Finance Communications**

<table>
<thead>
<tr>
<th>ICF Finance</th>
<th>Non-ICF Finance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports the following operations and functions:</td>
<td>Operations limited to the following operations and functions:</td>
</tr>
<tr>
<td>- Acquire</td>
<td>- Acquire</td>
</tr>
<tr>
<td>- Cancel</td>
<td>- Invite</td>
</tr>
<tr>
<td>- Cancel-invite</td>
<td>- Read</td>
</tr>
<tr>
<td>- End-of-group</td>
<td>- Read-from-invited-devices</td>
</tr>
<tr>
<td>- End-of-session</td>
<td>- Release</td>
</tr>
<tr>
<td>- Function-management-header</td>
<td>- Write</td>
</tr>
<tr>
<td>- Invite</td>
<td>- Write-with-invite</td>
</tr>
<tr>
<td>- Negative-response</td>
<td>Use of finance control byte allows function-management-header(^1) for the 3694 processor</td>
</tr>
<tr>
<td>- Release</td>
<td></td>
</tr>
<tr>
<td>- Read</td>
<td></td>
</tr>
<tr>
<td>- Read-from-invited-program-devices</td>
<td></td>
</tr>
<tr>
<td>- Timer</td>
<td></td>
</tr>
<tr>
<td>- Write</td>
<td></td>
</tr>
<tr>
<td>- Write-with-read</td>
<td></td>
</tr>
</tbody>
</table>

Communicates through ICF file

Supports requester (target) and acquired (source) sessions

Supports **chaining\(^2\)** or grouping of records

Verifies security information on the INIT-SELF request by using the AS/400 system

Uses ICF operations and functions to communicate between applications and devices

Notes:

1. A record that contains control information for the data that follows.

2. A method of storing records in which each record belongs to a list or group of records and has a linking field for tracing the chain.
Finance Communications
Network Example

Figure 1-2 is one example of an AS/400 finance communications network.

Finance Communications Using SNA Pass-Through Support

SNA pass-through is an AS/400 function that allows finance controllers and devices attached to the AS/400 system to communicate directly with a System/370 or System/390 host system. See “Configuring the AS/400 System for SNA Pass-Through Communications” on page 3-4 for information about configuring SNA pass-through support.
Chapter 2. Finance Communications Support

This chapter explains the functions supported by ICF and non-ICF finance communications interfaces.

Finance Functions

The types of finance controllers used with both ICF and non-ICF finance communications on the AS/400 system are:

- 3601 Finance Communications Controller
- 3694 Document Processor
- 4701 Finance Communications Controller
- 4702 Communications Processor

The types of finance controllers used only with ICF finance communications on the AS/400 system are:

- 4730 Personal Banking Machines
- 4731 Personal Banking Machines
- 4732 Personal Banking Machines
- 4736 Personal Banking Machines
- Financial Branch System Services (FBSS), including the 4737 Self-Service Transaction Station

The 3601 controller is the predecessor to the 4701 finance controller. Configure the 3601 as a 4701 controller with the appropriate exchange identifier (EXCHID) value.

4737 Self-Service Transaction Stations are configured as TYPE(*FBSS).

The 3694 processor is a programmable inscriber-reader-sorter that financial institutions use to inscribe, read, and sort magnetic-ink character recognition (MICR) documents and to capture data for subsequent host-system processing.

The 4701, 4702, and FBSS finance controllers are programmable and perform a variety of data processing transactions in a financial environment. FBSS is an IBM licensed program that provides extensions to the operating system of the personal computer or the PS/2® work station to support a finance industry environment. Through attached devices, tellers use finance communications for account inquiry and updating. Accountants use finance communications to maintain a record of cash flow through their businesses.

The 4730, 4731, 4732, and 4736 Personal Banking Machines are programmable and can dispense and accept money while keeping track of the transactions. Customers of financial institutions use Personal Banking Machines to transfer funds, determine account balance amounts, and make automatic payments.

The 4737 Self-Service Transaction Station is programmable and performs financial tasks such as transferring funds, determining account balance amounts, and making automatic payments.

Using finance communications, you can do the following:

- Send and receive data between the AS/400 system and the finance controller programs and describe how to present that data.
- Use SNA pass-through to provide communications between finance controllers and devices attached to the AS/400 system and a System/370 or System/390 host system.
- Communicate with 4701 and 4702 controllers using an X.21 circuit-switching network. X.21 is the Telecommunication Standardization Sector (CCITT), formerly known as ITU-T, specification that defines connection of data terminal equipment to an X.21 network.
- Communicate with 4701, 4702, and FBSS controllers through an X.25 packet-switching data network. X.25 is the ITU-T specification that defines the interface to an X.25 network. The AS/400 system attaches to an X.25 network using a nonswitched line through either an X.21 or X.21 bis (V.24 or V.35) physical shared boundary.
- Communicate with FBSS controllers on a token-ring network using Systems Network Architecture (SNA).
- Communicate between an FBSS controller on a token-ring network and an AS/400 system on an Ethernet network using the 8209 LAN bridge to connect the two LANs.
• Allow 3270 sessions and finance sessions to share the same X.25 virtual circuit.
• Allow attached 4701, 4702, and FBSS finance controllers to share the X.25 line with any AS/400 session types possible for X.25 communications.
• Attach any finance controller to your AS/400 system using the synchronous data link control (SDLC) protocol. SDLC is a type of communications line control that uses commands to control the transfer of data over a communications line. SDLC can have one of the following configurations:
  – Nonswitched point-to-point
  – Nonswitched multipoint
  – Switched point-to-point, using the following connection methods:
    - Automatic dial (uses a serial automatic dial modem).
    - Automatic call (two-line EIA 366/V.25) support. You need RPQ 843567 for the 9406 System Unit and RPQ 843568 for the 9404 System Unit.
    - Automatic answer.
    - Manual dial.
    - Manual answer.

The finance controllers can share the same SDLC multipoint communications line with any other session type using an SDLC primary role on the AS/400 system.

Note: The AS/400 system allows a maximum of 254 controllers to be attached to a multipoint line.

Concurrent Sessions

The AS/400 system supports concurrent operation of 3270 displays, printers, and finance work stations on one controller. However, the controller must also support concurrent LU0 and 3270 (LU1, LU2, and LU3) device operation. For 4701 and 4702 controllers, the type and the application of each device attached to the controller must be defined in the controller program generator (CPGEN) file on the controller. The CPGEN file determines what session type the device will be using.

The 3270 device support includes the IBM 3277, 3278, and 3279 displays and the IBM 3287 printer. For more information about the 3270 workstation support, see the topic on using the 3270 remote attachment support in the Remote Work Station Support, SC41-5402.

Security

The security provided on the AS/400 system controls the use of communications device descriptions and commands used with the device descriptions. Security also controls access to programs and objects used by programs.

For finance-specific considerations, see Chapter 7, “Finance Considerations,” and for general system security information, see the Security – Reference book.

Error Handling and Recovery

Application programs use major and minor error return codes to handle error conditions. Applications written in high-level languages can access return codes to help diagnose problems. The C Set ++ for OS/400, ILE COBOL/400, and ILE RPG/400 languages provide language-defined file values for status that can be used together with the major and minor codes. Messages are entered in the job log to identify the error that occurred.

Note: The ILE C/400 language does not support file status values. However, you can look at the value of the global variable in the errno.h header file to check for any I/O exceptions that may occur and to retrieve any exception data associated with the error.

If an error results in a notify message, it is recoverable with little or no operator involvement.

For error recovery specific to ICF finance, see Appendix B, “Return Codes, Messages, and Sense Codes.” For error recovery specific to non-ICF finance, see Chapter 6, “Writing Non-Intersystem Communications Function Application Programs.”

For information describing recovery from line errors, see the Communications Management book.
System Monitor Session

The system monitor is a controller program used to perform service, configuration, and debugging functions on 4701 and 4702 finance controllers. Local location address (LOCADR) 01 is reserved for this program. When using finance communications, your ICF application program communicates with the system monitor program by using a device configuration specified as LOCADR (01) and a finance type of *FNCICF.

4701 Finance Controller Diskette

Download: The AS/400 system supports the downloading of an operating diskette image for an 8-inch diskette. The operating diskette image for the 4701 finance controller must be created on a System/370 computer and loaded on the AS/400 system. After the diskette image is loaded on the AS/400 system, use the diskette download support to send the diskette image to the controller through the system monitor session. For more information about this feature, see Appendix D, “4701 Finance Controller Diskette Download.”

Finance Interfaces

Operating System/400* (OS/400*) finance communications provides support allowing application programs to communicate with finance controllers. The application programmer can use ICF finance or non-ICF finance to access this support.

Regardless of the interface you choose, you must supply the following:

- Finance controller application programs
- AS/400 application programs to process financial transactions

You can write application programs on the AS/400 system using the supported high-level language licensed programs.

Intersystem Communications Function Finance Support

Specify TYPE(*FNCICF) in the device description to identify that you are using ICF finance. Application programs use an ICF file to send and receive data. The ICF file contains the file description identifying the record formats used by the application programs. For more information about writing ICF finance applications, see Chapter 5, “Writing Intersystem Communications Function Finance Application Programs.”

Non-Intersystem Communications Function Finance Support

The non-ICF finance communications support is separate from and completely independent of ICF finance communications support. Specify 4704, 3624, or 3694 for the TYPE parameter in the device description to identify that you are using non-ICF finance. Non-ICF finance uses a display file to send and receive data. The display file contains the file description identifying the record formats used by the application program and finance support.

You can choose one of the following for non-ICF finance communications:

- The Submit Finance Job (SBMFNCJOB) command with the Work with Table (WRKDEVTBL, WRKUSRTBL, or WRKPGMTBL) commands.
  
  For information about the Submit Finance Job and Work with Table commands, see “Using the Submit Finance Job Command” on page 6-1.

- Finance I/O Manager (FIOM) alone or with either the SBMFNCJOB command or the user-defined data stream (UDDS). FIOM is the recommended alternative to UDDS. For more information about FIOM, see “Using Finance Input/Output Manager” on page 6-7.

- UDDS. For more information about UDDS, see “Using User-Defined Data Stream” on page 6-14.
Chapter 3. Configuring Finance Support

This chapter describes the commands used to configure your AS/400 system for finance communications. See the Communications Configuration book for a complete description of AS/400 configuration commands and parameters.

Configuring the AS/400 System for Finance Communications

The following commands allow you to create, change, display, print, and delete the line, controller, and device descriptions.

Configuring the Line Description

An AS/400 line description contains configuration information, such as the port number for communications lines, network characteristics, and physical information needed by the system. Create a line description for each communications line used to communicate with a finance controller.

Use the following commands to create, change, display, print, or delete line descriptions:

CRTLINS DLC
The Create Line Description (SDLC) command allows you to create a line description for each SDLC communications line used to communicate with the finance controllers. Each line should be created as the SDLC primary role.

CRTLNX25
The Create Line Description (X.25) command allows you to create a line description for each X.25 communications line used to communicate with the 4701, 4702, and Financial Branch System Services (FBSS) controllers.

CRTLINTRN
The Create Line Description (Token-ring) command allows you to create a line description for a token-ring network used to communicate with finance controllers.

CRTLINETH
The Create Line (Ethernet) command allows you to create a line description for an Ethernet line used to communicate with finance controllers over an 8209 LAN Bridge that connects the Ethernet line to a token-ring LAN to which the controller is attached.

CHGLINS DLC
The Change Line Description (SDLC) command allows you to change the configuration parameters for an SDLC communications line used to communicate with finance controllers.

CHGLNX25
The Change Line Description (X.25) command allows you to change the configuration parameters for an X.25 communications line used to communicate with finance controllers.

CHGLINTRN
The Change Line Description (Token-ring) command allows you to change the configuration parameters for a token-ring network used to communicate with finance controllers.

CHGLINETH
The Change Line Description (Ethernet) command allows you to change the configuration parameters for an Ethernet line used to communicate with finance controllers over an 8209 LAN Bridge that connects the Ethernet line to a token-ring LAN to which the controller is attached.

DSPLIND
The Display Line Description command allows you to display or print a line description.

DLTLIND
The Delete Line Description command allows you to delete a line description.

Configuring the Controller Description

A finance controller description defines configuration parameters unique to each finance controller, such as the station address to the system.

These controller descriptions are linked to the configured nonswitched line description when you
create the descriptions. Controller descriptions for switched connections are linked to the first available line in the configured switched line list when the call is made.

Use the following commands to create, change, display, or delete finance controller descriptions:

**CRTCTLFNC**
The Create Controller Description (Finance) command allows you to create a finance controller description for each finance controller with which your AS/400 system is communicating.

**CHGCTLFNC**
The Change Controller Description (Finance) command allows you to change the configuration parameters for a finance controller description.

**DSPCTLD**
The Display Controller Description (Finance) command allows you to display or print a controller description.

**DLTCTLD**
The Delete Controller Description (Finance) command allows you to delete a controller description.

### Configuring the Device Description

Device descriptions contain the logical unit (LU) address and device type of each device on a finance controller on the AS/400 system. (The LU address is known as the local location address.)

The local location address must match the address you used to identify a particular physical device. The local location address (LOCADR) parameter you specify can be a value from hex 02 through 3C for the non-ICF finance communications method and hex 01 through FF for ICF finance.

The device description also defines which finance communications method, ICF or non-ICF, is used. If you use ICF with a finance controller, you can also define a system monitor session by specifying a local location address of 01 and a finance type of *FNCICF.

You can create device descriptions for the following finance types:

- ICF finance device description (*FNCICF)
- 3624 Consumer Transaction Facilities (CTF)
- 3694 Document Processors
- 4704 Finance Communication Display Stations

**Note:** When you define a device as *FNCICF, the device can be any supported finance device.

Use the following commands to create, change, display, print, or delete device descriptions:

**CRTDEVFNC**
The Create Device Description (Finance) command allows you to create a device description for a finance device that will be linked to a finance controller. If you use ICF finance communications, you must specify *FNCICF as the device type (TYPE) parameter in the CRTDEVFNC command. If you use non-ICF finance, specify the specific device number, such as 4704, 3624, and 3694, in the TYPE parameter.

**CRTDEVDSP**
The Create Device Description (Display) command allows you to create a device description for each 3270 family display linked to each finance controller. Specify the class as a remote (*RMT) device. You can attach 3277, 3278, 3279, and 3287 device types to a finance controller.

**CRTDEVPRT**
The Create Device Description (Printer) command allows you to create a device description for each 3287 printer attached to the finance controller. Specify the class as a remote (*RMT) device.

**CHGDEVFNC**
The Change Device Description (Finance) command allows you to change the configuration parameters for each finance controller.

**CHGDEVDSP**
The Change Device Description (Display) command allows you to change the configuration parameters for each display attached to the finance controller.

**CHGDEVPRT**
The Change Device Description (Printer) command allows you to change the configuration parameters for each printer attached to the finance device.
The Display Device Description command allows you to display or print a device description.

The Delete Device Description command allows you to delete a device description.

You can define up to the following maximum number of devices for each controller:

<table>
<thead>
<tr>
<th>Controller</th>
<th>Maximum Devices</th>
<th>Maximum Active at Once</th>
</tr>
</thead>
<tbody>
<tr>
<td>4701</td>
<td>255</td>
<td>120</td>
</tr>
<tr>
<td>4702</td>
<td>255</td>
<td>120</td>
</tr>
<tr>
<td>FBSS</td>
<td>255</td>
<td>240</td>
</tr>
<tr>
<td>4731, 4732, 4736</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3694</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4730</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Only devices specified as TYPE(3694) or TYPE(*FNCICF) can be attached to the 3694 controller. Only devices specified as TYPE(*FNCICF) can be attached to the 4730, 4731, 4732, or 4736 controllers. An FBSS controller can have *FNCICF, 3277, 3278, 3279, and 3287 devices attached. You can configure any of the following device types for a 4701 or 4702 controller: 3277, 3278, 3279, 3287, 3624, 4704, and *FNCICF.

**Note:** If you are using non-ICF finance communications, you can attach a 4710 or 4720 printer to the AS/400 system by configuring the printer as a 4704 device. If you are using ICF finance communications, configure the printer as a *FNCICF type.

The device type you specify in the device description determines the format of the data stream sent to the finance controller. The data stream can be one of the following:

- LU0 for device type 3624, 3694, 4704, or *FNCICIF
- LU1 for device type 3287
- LU2 for device type 3277, 3278, or 3279

**Note:** If you specify a 32XX device, the 4701, 4702, and FBSS controllers must be capable of supporting 3270 logical unit 1 (LU1) and 3270 logical unit 2 (LU2) data streams.

### Using 3270 Devices

If your 4701, 4702, or FBSS finance controller can support the 3270 family of displays and printers, you can also attach those devices to your finance controller.

The AS/400 system requires that any device configured as a 3278 or a 3279 must accept extended data streams. Be sure that the application program running on the 4701, 4702, or FBSS controller has extended data stream support. If it does not have extended data stream support, configure each device as a type 3277. LU0 data streams allow printer data to be included with display data. LU1 data streams are sent to printers. LU2 data streams require you to create 3270-type application displays. Be aware that translation and emulation can make the LU2 approach run slower than the LU0 support.

You can program a 4701, 4702, or FBSS controller to do 3270 emulation. You can also configure a finance device as a 3270 display, or you can attach 3278, 3279, and 3287 devices to the controller. You must use an LU2 emulation package on the controller regardless of which devices you use. Using this mode of operation, you can configure the controller as follows:

- As an FBSS controller with 3270 emulation or 3287 printer emulation plus FBSS-supported finance devices attached
- As a 4701 or 4702 controller with 3277, 3278, 3279, 3287 devices and, optionally, 3624 and 4704 devices attached
- As a 3274 controller with only 3270 devices attached

For more information about 3270 remote attachment support, see the *Remote Work Station Support* SC41-5402 book.

You must vary on the line, controller, and device descriptions for finance support before any AS/400 application program can communicate with a finance controller. For information about the vary on process, see Chapter 4, “Running Finance Support.” For more information about the communications configuration process and the commands used for configuration, see the *Communications Configuration* book.
Configuring the AS/400 System for SNA Pass-Through Communications

SNA pass-through support allows finance controllers and devices attached to an AS/400 system to communicate directly with a System/370 or System/390 host system using the SNA LU0, LU1, or LU2 protocols.

You configure the AS/400 system for SNA pass-through communications by creating line, controller, and device descriptions. These objects describe the connections between the AS/400 system and the host system and between the AS/400 system and the finance controller and devices.

Line Descriptions

Two line descriptions must be created, one describing the connection between the finance controller and the AS/400 system, the other describing the connection between the AS/400 system and the host system. The line descriptions need not be of the same type; for example, the finance controller can be attached to the AS/400 system using a token-ring line while the AS/400 system is connected to the host system using an SDLC line.

The following commands may be used to create the line descriptions:

- **CRTLINIDLC**
- **CRTLINETH**
- **CRTLINSDLC**
- **CRTLINTRN**
- **CRTLINX25**

Notes:

1. ISDN communications (using IDLC or X.25 lines) requires additional configuration of a network interface description and connection list. See the ISDN Support book for more information and examples of ISDN communications.

2. An FBSS controller attached to a token-ring network can be connected to an AS/400 system on an Ethernet network by using the 8209 LAN Bridge. The LAN and Frame Relay Support book contains configuration examples and considerations for bridged environments.

Controller Descriptions

Two controller descriptions must be created for SNA pass-through communications. A host controller, describing the host system, must be created using the CRTCTLHOST command. An APPC controller must also be created to represent the finance controller that will communicate with the host system. Use the CRTCTLAPPC command to create the APPC controller description.

For finance controller types 3694, 4701, 4702, 4731, 4732 or 4736, set the TYPE parameter on the CRTCTLAPPC command to the controller type. Specify *NO for the APPN parameter to indicate that the controller does not use APPN—LU 6.2. For instance, for a 3694 controller, specify: TYPE(3694) APPN(*NO).

For a PS/2* work station running FBSS with the DOS operating system, specify: TYPE(*FBSS) for APPC controllers.

For a PS/2* work station running FBSS with the OS/2* operating system, specify either APPN(*YES) or TYPE(*BLANK). If you specify TYPE(*BLANK), you must set the SSCPID parameter to a value other than zero.

Device Descriptions

At least two device descriptions must be created; a finance device and an SNA pass-through device. The SNA pass-through device description associated with the host logical unit is created using the CRTDEVSNPT command and must specify SNPTCLS(*UP). The finance device description is created using the CRTDEVFNC command and must specify FNCCLS(*SNPT) and TYPE(*FNCICF).

3270 device descriptions attached to finance controllers can also be configured as SNA pass-through devices using the CRTDEVDSP and CRTDEVPRT commands.

Note: Finance controllers, attached finance devices, and attached 3270 devices that are configured to use SNA pass-through support must be reconfigured to allow communications between the
finance controller and the AS/400 system. To reconfigure for communications between the finance controller and the AS/400 system, you must vary off the configuration objects associated with the finance controller (line description, APPC controller, and devices) and reconfigure the controller using the CRTCTLFNC command. If the connection uses a switched SDLC line or X.25 communications, you must also delete the APPC controller description.

**Associating Finance and SNA Pass-Through Devices**

SNA pass-through configuration requires that each finance device be paired with an SNA pass-through device. The logical association of the devices can be done in either of two ways:

- **Specify the SNA pass-through device name:** The SNA pass-through device (SNPTDEV parameter) on each device description can specify the name of the other device description. That is, the finance device description SNPTDEV parameter specifies the name of the SNA pass-through device; the SNA pass-through device SNPTDEV parameter specifies the name of the finance device.

- **Specify a group of SNA pass-through devices:** The SNA pass-through group name (SNPTGRP parameter) on each device description can specify the name of a group of devices. For example, the finance device description can specify the name of a group of SNA pass-through devices. SNA pass-through groups are defined in configuration list entries for the QSNAPASTHR configuration list.

This method allows the system to associate the finance device with any SNA pass-through device defined in the group. When the finance device is varied on, the system will select the first available SNA pass-through device listed in the group specified on the SNPTGRP parameter.

Only one configuration list of type *QSNAPASTHR is allowed on the system at one time. Use the Add Configuration List Entries (ADDCFGLE) command to define SNA pass-through groups within the QSNAPASTHR configuration list.

A finance device that specifies DEVCLS(*SNPT), SNPTDEV(*NONE), and SNPTGRP(*NONE) can be associated with an SNA pass-through device only if the SNA pass-through device either specifies the name of the finance device on the SNPTDEV parameter or specifies an SNA pass-through group (SNPTGRP parameter) that includes the finance device.

See the book, *Remote Work Station Support, SC41-5402* for more information and examples of configuring SNA pass-through support.

**Using the Work with Table Commands to Configure Non-Intersystem Communications Function (Non-ICF) Finance**

You can use the following commands to configure the Submit Finance Job (SBMFNCJOB) command for non-ICF finance communications. For more information about the syntax or parameters for these commands, see the *CL Reference* book.

**Work with Device Table Command**

A finance device table is a list of finance devices that can be acquired by the finance job using the SBMFNCJOB command. Use the Work with Device Table (WRKDEVTBL) command to create, change, delete, or display finance device tables.

Device table updates can be used by any finance job that is submitted after all changes are done.

When you enter the WRKDEVTBL command, the Work with Device Tables display appears:

![Work with Device Tables](image)

**Chapter 3. Configuring Finance Support**
Create a new table
Create one new table at a time by typing a 1 in the first blank line under the Opt column and typing in the new name.

Change a table
Select a table to change by typing a 2 next to the table name in the list, or by typing a 2 in the first blank line under the Opt column and typing in the name.

Delete one or more tables
Type a 4 next to each table you want to delete.

Display a table
Type a 5 next to the name of the table you want to display.

The Create Device Table display appears when you select option 1 (Create) on the Work with Device Tables display:

```
Create Device Table
Device table . . . . . : BANKDEV4
Type information, press Enter.
Text . . . . . . . . . .
```

To add a device name, enter the device on one of the input lines supplied. The device name you add should have a device description (created using the CRTDEVFVCN command) with the same name (DEV parameter) and a device type of 3624 or 4704. Messages appear warning you that the device description does not yet exist, or that the device type is not 3624 or 4704, but the device name is still added to the table. You can also specify up to 50 characters of descriptive text for the new table on the line supplied.

Work with User Table Command

The finance user table is a list of approved finance user IDs. The finance job uses these IDs to approve user IDs sent in the data stream with the INIT-SELF request from the finance controller. Finance user IDs are not related to user profiles.

Use the Work with User Table (WRKUSRTBL) command to create, change, delete, or display any number of user tables that define the finance user IDs. Each table must have a unique name.

User table updates can be used by any finance job that is submitted after all changes are done.

When you enter the WRKUSRTBL command, the Work with User Tables display appears:

```
Work with User Tables
Position to . . . . .
Type options [add Program Table], press Enter.
1=Create 2=Change 4=Delete 5=Display
Opt User Table Text
· TESTUSR1 third sample user table
· TESTUSR2 another sample table
· USER1 Simple user table for IMPRJOB command
· USER2 second sample user table
```

Create a new table
Create one new table at a time by typing a 1 in the first blank line under the Opt column and typing in the new name.

Change a table
Select a table to change by typing a 2 next to the table name in the list, or by typing a 2 in the first blank line under the Opt column and typing in the name.

Delete one or more tables
Type a 4 next to each table you want to delete.

Display a table
Type a 5 next to the name of the table you want to display.

The Create User Table display appears when you select option 1 (Create) on the Work with User Tables display:

```
Create User Table
User table . . . . . : TESTUSR1
Type information, press Enter.
Text . . . . . . . . . .
```

To add a user table, enter the user ID on one of the input lines supplied. You can specify up to 50 characters of descriptive text for the new table on the line supplied.
Work with Program Table Command

The finance program table is a list of your AS/400 transaction-processing application programs, with their associated program IDs. These program IDs are included with the data in the transaction request by a finance controller. The program ID is then used to call the correct AS/400 application program to process the transaction.

Use the Work with Program Table (WRKPGMTBL) command to create, change, delete, or display, any number of program tables. Each table must have a unique name.

Program table updates can be used by any finance job that is submitted after all changes are done.

When you enter the WRKPGMTBL command, the Work with Program Tables display appears:

Create a new table
Create one new table at a time by typing a 1 in the first blank line under the Opt column and typing in the new name.

Change a table
Select a table to change by typing a 2 next to the table name in the list, or by typing a 2 in the first blank line under the Opt column and typing in the name.

Delete one or more tables
Type a 4 next to each table you want to delete.

Display a table
Type a 5 next to the name of the table you want to display.

The Create Program Table display appears when you select option 1 (Create) on the Work with Program Tables display:

Enter new program IDs and program names by typing the following information on one of the extra input lines supplied:

- PROGRAM ID (required)
- PROGRAM NAME (required)
- LIBRARY NAME (optional - c5197LIBL is default)

A message appears warning that the program or the library does not exist. However, entries are still added to the table. You can also specify up to 50 characters of descriptive text for the new table on the line supplied.
Chapter 4. Running Finance Support

This chapter contains information you need to run your network, including information about the Vary Configuration (VRYCFG) command and the Work with Configuration Status (WRKCFGSTS) command.

Varying Finance On and Off

You must define and vary on the communications configurations before using your communications applications. You can specify the configurations to be automatically varied on at initial program load (IPL) when you create your configurations. You can also use the Vary Configuration (VRYCFG) command or the Work with Configuration Status (WRKCFGSTS) command to vary on and off the appropriate network interface line, controller, and device descriptions.

Using the Vary On Configuration Command

When using the VRYCFG command, specify the following parameters:

**CFGOBJ**
Specifies the name of the line, controller, or device description to be varied on or off, or a list of names of configuration objects of the same description type (for example, a list of line description names).

**CFGTYPE**
Specifies the type of configuration to be varied on or off. This is a required parameter. Valid entries are:
- *CTL*: Controller configuration
- *DEV*: Device configuration
- *LIN*: Line configuration

**STATUS**
Specifies the status of the configuration object. Valid entries are:
- *ON*: Object is varied on.
- *OFF*: Object is varied off.

**Note:** All devices must be varied off before the attached controller can be varied off. All controllers must be varied off before the associated line can be varied off. (This can be done by using the RANGE parameter.) A device can be varied off only when it is not allocated for an active job.

**RANGE**
Specifies what group of configuration objects should be varied on or off. Valid entries are:
- *NET*: All downline configuration objects are varied on or off. Downline objects are devices attached to a controller and controllers that are attached to a communications line in a communications configuration.
- *OBJ*: Only the specified object is varied on or off.

**Note:** When *NET is specified, the system does the vary off procedures in the appropriate order. The objects include the configuration object or objects specified and their attached configuration objects only. For example, for a communications line, the attached objects are controllers and devices; for a controller, the attached objects are devices. Devices do not have attached objects.

**VRYWAIT**
Specifies whether the Ethernet, token-ring, X.25, or switched SDLC line description is varied on asynchronously or synchronously. The VRYWAIT parameter specifies how long the system must wait for synchronous vary on to be completed. Once completed, the ICF file can be opened and the session acquired.

- *CFGOBJ*: Specifies that the VRYWAIT parameter value on the line description is to be used.
- *NOWAIT*: Specifies that the line is to be varied on asynchronously. This signals the system not to wait for vary on completion.
• vary-on-wait: Specifies a value ranging from 15 through 180 seconds in 1-second intervals. The system waits until either the line is varied on or the timer expires before completing the VRYCFG command.

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

Using the Work with Configuration Status Command

Using this command, you access the Work with Configuration Status display.

On the Work with Configuration Status display, you can choose the following options:
• Vary on (option 1): Varies on the line, controller, or device and all of the attached controllers or devices. This is the same as using the Vary Configuration (VRYCFG) command with STATUS(*ON).
• Vary off (option 2): Varies off the line, controller, or device and all of the attached controllers or devices. This is the same as using the Vary Configuration (VRYCFG) command with STATUS(*OFF). You may vary off devices only if they are not allocated to an active job. Jobs can be canceled if you need to vary off a device.

For the syntax of the VRYCFG and WRKCFGSTS commands, see the CL Reference book. For more information about the VRYCFG and WRKCFGSTS commands, see the Communications Management book.

Defining Communications Entries

IBM supplies two subsystem configurations to support ICF finance communications. These are QBASE, the controlling subsystem, and QCMN, the communications subsystem. The controlling subsystem is the interactive subsystem through which the system operator controls the system.

The communications subsystem is used when QCTL is the controlling subsystem. If either of these subsystems is active, program start requests may be accepted for finance communications.

The AS/400 system considers communications controllers to be another source of work for the subsystem. Therefore, you must use a communications entry in the subsystem description to identify the devices from which work (the program start request) can be received by the subsystem. To create subsystem descriptions using the Create Subsystem Description (CRTSBSD) command, see the CL Reference book.

Default communications entries are shipped with the subsystem. QBASE and QCMN subsystem descriptions are shipped with a default communications entry specified as DEV(*ALL) and MODE(*ANY) to allow program start requests from any ICF communications type. With the following commands, you can change these entries:
• Add Communications Entry (ADDCMNE):
  Adds a device or devices to the subsystem.
• Remove Communications Entry (RMVCMNE):
  Removes a device or devices from a subsystem.
• Change Communications Entry (CHGCMNE):
  Changes an existing entry for a device.

Note: These commands cannot change the communications entries of the subsystem if the subsystem is active.

For more information about these commands, see the Communications Management book. For more information about managing your subsystems, see the Work Management book.

Starting the Finance Subsystem

The appropriate subsystem must be started if you want to use finance devices.

Non-Intersystem Communications Function: For non-ICF finance, the finance subsystem (QFNC) must be started by using the Start Subsystem (STRSBS) command before you can use the Submit Finance Job (SBMFNCJOB) command.
**Intersystem Communications Function**

**Finance:** For ICF finance, the appropriate sub-system must be started before program start requests can be received from the finance controller. Use the Start Subsystem (STRSBS) command to start a subsystem to be used for ICF finance communications.
Chapter 5. Writing Intersystem Communications Function Finance Application Programs

This chapter describes how an application program uses the AS/400 system, intersystem communications function (ICF) support, and finance communications to communicate with a finance controller. You can code the program using supported, high-level language programs to do the following functions:

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

Notes:

1. Check the major and minor return codes, as well as any response indicators, after each operation that your program issues.

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 the data description specifications (DDS) keywords or system-supplied formats. For a list of DDS keywords and system-supplied formats, see Appendix A, “Language Operations, DDS Keywords, and System-Supplied Formats,” or the DDS Reference book.

To help you write and use programs on the AS/400 system, see Appendix E, “Intersystem Communications Function Finance Example Programs.”

Using Intersystem Communications Function File Commands

An Operating System/400 intersystem communications function (ICF) file must be created before your application can use finance communications. For more information about the ICF file, see the book, ICF Programming.

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

The following commands are valid for finance communications:

**CRTICFF**

The Create ICF File command allows you to create an ICF file specifying file level attributes.

**CHGICFF**

The Change ICF File command allows you to change the file attributes of the ICF file.

**OVRICFF**

The Override ICF File command allows you to temporarily change the file attributes of the ICF file at run time. These changes are in effect only for the duration of the job and do not affect other users of the file.

**DLTOVR**

The Delete Override command deletes the effect of the OVRICFF command.

**DLTF**

The Delete File command deletes the file from the system and frees the storage space allocated to that file.

**DSPFD**

The Display File Description command displays information about the attributes of the device file.

**DSPFFD**

The Display File Field Description command displays field-level information for a device file.

**ADDICFDEVE**

The Add ICF Device Entry command adds a program device entry, with the specified device name and attributes, to the file. You can use this command many times to...
add multiple program devices to the same file.

**CHGICFDEVE**
The Change ICF Device Entry command changes the program device entry defined with the ADDICFDEVE command.

**OVRICFDEVE**
The Override ICF Device Entry command overrides the attribute specified in the ADDICFDEVE command, or temporarily associates the specified program device name and attributes with the file.

This command differs from the ADDICFDEVE command in that it does not permanently change the ICF file. The association between the program device entry and the file exists only for the job in which the command runs. You can use this command many times to override multiple program device entries in the file.

**DLTOVRDEVE**
The Delete Override Device Entry command deletes the effect of the OVRICFDEVE command.

**RMVICFDEVE**
The Remove ICF Device Entry command removes one or more program device entries from the file.

**DSPOVR**
The Display Override command displays overrides created by the OVRICF file command.

The program device entry associates a program device name with a device description.

The ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands use the following parameters for finance communications:

**FMTSLT**
Determines the format selection option. This parameter indicates the type of processing used to determine what record format to use on input operations. Finance communications supports only the program (*PGM) and record identification (*RECID) values of the FMTSLT parameter. For more information about the FMTSLT parameter, see the book, *ICF Programming*.

**PGMDEV**
Specifies the program device name being defined (the name used by the program to do the operations). The program device name must be unique throughout all entries in the file. You can map two or more different program device names to the same communications configurations.

PGMDEV is a required parameter.

**RMTLOCNAME**
Specifies the remote location name associated with the program device. The remote location name specifies another system with which your system can communicate in an SNA network. A remote location is associated with any device description that contains the same remote location name.

You cannot configure the same remote location name in more than one device description.

The system uses the remote location name to select the device description. The RMTLOCNAME parameter does not need to exist at the time you define the program device entry; however, it must exist as a part of the device description on the system when the program is acquired. If your program is started by a program start request, a RMTLOCNAME of *REQUESTER must be used.

RMTLOCNAME is a required parameter.

---

**Starting a Session**

Before your program can communicate with the controller program, you must establish an ICF communications session. A communications session is a logical connection between two systems through which a finance controller program communicates with a program on an AS/400 system. This communications session is established with an open/acquire operation and is ended with a release operation or an end-of-session function or a close operation.
Starting a Session with a 47xx or Financial Branch System Services Controller

You can start the session with the 47xx or FBSS controllers in one of two ways: by the controller sending an INIT-SELF command and then sending a program start request, or by your AS/400 program issuing an ICF Acquire operation.

The following explains how to start a controller-initiated session:

1. Start the SNA session from the controller by sending an INIT-SELF command (considered a logon request). The AS/400 system sends a BIND command to start the SNA session. On a secure AS/400 system, the BIND command is sent only if the user data field of the received INIT-SELF command contains a valid user ID and password.

   The user data field may also contain a default library name. If the library name field is included, that library is used to search for a program that is specified in a program start request sent later by the controller (if a specific library is not specified on that request).

2. Start the ICF session and transaction by sending a program start request using an *EXEC statement. The AS/400 finance support starts the program specified on the *EXEC statement. The target program issues the ICF Acquire operation and establishes the session and transaction.

The following explains how to start an AS/400-initiated session:

1. Start the SNA session from your AS/400 source program by issuing an ICF Acquire operation. This is known as an unsolicited start session request because the controller does not request the session.

2. The session and transaction are established automatically.

Starting Your Program: Your finance programs can be started by the controller sending a program start request or by the use of a manually started program.

Program Start Request from the Controller:
Typically, your program is started after a program start request is received from the controller. The program start request contains an *EXEC request statement with a program name, an optional library name, and optional program parameters.

Note: Program start requests can be received after the AS/400 system sends a BIND command and before the AS/400 system sends an UNBIND command.

Additional program start requests received while a transaction program is running are treated as user data.

The syntax of the program start request is:

*EXEC program name[,library name] [optional data]

Program name
Name (from 1 to 10 characters in length) of the program to be started. The program name must start in position 7. A blank must be in position 6.

Library name
Name (from 1 to 10 characters in length) of the library containing the program to be started. This parameter is optional. If this name is specified on both the INIT-SELF command and the *EXEC request statement, this library name is used for the program start request. If the library name is not specified in either place, the program must exist in a library of the job description specified in the communications entry of the subsystem in which it runs.

Optional data
From 1 to 512 bytes of optional data following the program name or an optional library name. A blank must be included between the program name, or the optional library name, and the data.

AS/400 System Started Program: You can start the program with an AS/400 command or a command from another program running on the AS/400 system.
Starting a Session with a 3694 Document Processor

For a 3694 processor, the session and program are both started as a result of a program start request, which is an INIT-SELF request sent by the controller. On a secure AS/400 system, this INIT-SELF must also include information for determining a user ID and password.

Note: ICF finance supports only target programs for the 3694 processor. The 3694 must always establish the session to start the program on the AS/400 system. The AS/400 system cannot initiate a session on the controller.

If the 3694 processor sends optional data in the user data field, there must be a minimum of 10 bytes of data in the following format:

<table>
<thead>
<tr>
<th>User Data Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Request code.</td>
</tr>
<tr>
<td>2 through 4</td>
<td>User ID; these 3 characters are added to USER to form the user identifier.</td>
</tr>
<tr>
<td>5 through 8</td>
<td>Password.</td>
</tr>
<tr>
<td>9 through 10</td>
<td>These 2 characters are added to LIB to form the library name. If these characters are 00, the current library list of the sub-system containing the communications entry is used.</td>
</tr>
</tbody>
</table>

The name of the program to be started is specified in the destination logical unit (DLU) field of the INIT-SELF command. For more information about the INIT-SELF command format, see Figure 7-2 on page 7-7.

Open/Acquire Operation

Communications sessions using ICF finance are always started with an acquire operation. Before any input or output operations can be directed to a program device, the program device must be acquired. Your application program uses the acquire operation to establish a session between your program and the finance controller. To be acquired, program devices must be defined to the ICF file by using one of the following commands:

- Add ICF Device Entry (ADDICFDEVE)
- Override ICF Device Entry (OVRICFDEVE)

A program device can be implicitly acquired through the open operation or explicitly acquired through the acquire operation. The acquire operation can be used many times with different program device names. When you explicitly acquire a device using the acquire operation, you identify the session you want to establish by using the same program device name on the acquire operation as you specified on the PGMDEV parameter for the ADDICFDEVE or the OVRICFDEVE command.

You must specify the RMTLOCNAME parameter on the Add or Override ICF Device Entry command. If the session is a controller-initiated session, you must specify *REQUESTER for the RMTLOCNAME parameter. If the session is an AS/400 system-initiated session, you must assign a value for the RMTLOCNAME parameter. The parameter must match the remote location name in the device description with which you want to communicate. For more information about starting your program, see “Starting Your Program” on page 5-3.

You can use the wait file (WAITFILE) parameter of the CRTICFF, OVRICFF, or CHGICFF command to specify the maximum amount of time you want to wait for session resources to become available.

Note: The WAITFILE value is not used for sessions where the connection to the controller is over a switched connection, for example, a SDLC switched line, a X.25 SVC, Ethernet, or a Token-Ring connection.

Sending Data

Data is sent to and received from the controller by using write, read, and write-with-read operations. The write operations are done using a record format. The results of these operations are sent to the program by using major and minor codes, high-level status values, and the input/output feedback area.

Finance communications buffers data. The first record and the last record of a group of records are sent to the controller prior to the completion of the output operation. However, output operations for records that are not the first or last in a group...
of records are buffered by finance communications. These records are sent when the buffer becomes full, or when the last record of the group is sent.

Write Operation

The write operation passes one or more data records from the AS/400 program to the remote controller program in this session. Finance communications determines the size of the record from the MAXLENRU parameter in the device description and will block your data into as many records as needed.

Note: Finance communications does not support multiple record groups for the 3694 processor.

Finance communications supports write operations while the session is invited. If the write operation ends successfully, and if the invite function is not specified on the write operation, the session is no longer invited when the write operation ends. If the write operation does not end successfully, or if the invite function was specified on the write operation, the session remains invited.

Force-Data Function

Your program uses the force-data function to send immediately (without waiting for the buffer to become full) any data currently being held in the buffer. Any data specified on the same operation as the force-data request is also sent. If data is not specified on a write operation with the force-data function, a null record is sent.

For more information about buffering data, see “Buffering” on page 7-1.

End-Of-Group Function

Use the end-of-group (ENDGRP) function to indicate the end of a user-defined group of records. When the ENDGRP function is issued, the control of the session goes back to a contention state. Then, either the AS/400 system program or the controller program can send data. When the ENDGRP function is used, finance communications requires a response from the finance controller to the group just sent. Control is not returned to the application program until a response is received.

Note: This function does not apply for a 3694 controller.

Function-Management-Header Function

Use the function-management-header function to send control information to the controller program about the data following the function-management-header. You must designate data on a write operation that specifies the function-management-header function. Your program is responsible for creating the function-management-header.

Receiving Data

You can use the read and read-from-invited-program-device operations to receive data. When your program receives data with a read or read-from-invited-program-devices operation, all the records of the group are given to your program. A read operation does not end until the entire group of records is received by the finance communications support.

For 47xx and FBSS controllers, the read operation also closes a partially sent group of records. If a group of records is implicitly closed by a read operation, a response from the controller program to the group of records being sent is not required.

Read Operation

Your program uses the read operation to obtain data from the controller. This operation causes finance communications to obtain data from the controller program with which your program is communicating. The read operation causes your program to wait for the data if it is not immediately available. Your program receives control when the data is available. The READ operation waits until a complete group of records is received. If an SNA Cancel command is received, the data is discarded and the read operation waits for the next complete group of records.

Finance communications always waits until an entire group of records is received before returning control to your program. If the length of the field in the record format you are using is large enough, the maximum data length that can be
received is the value you specified in the MAXRCDLEN parameter of the Create ICF File (CRTICFF), Change ICF File (CHGICFF), and Override ICF File (OVRICFF) commands. If the format you are using is not large enough, you receive an error return code.

You do not need to issue an invite function before a read operation to receive data. However, if a program device has an outstanding invite to which a read is issued, the read completes the invite and receives the data.

**Note:** The read operation obtains data from a specified program device while the read-from-invited-program-devices allows data to come from any previously invited program device.

When a group of records is partially sent by your program, a read operation or a write-with-read operation that does not explicitly specify the end-of-group function implicitly sends the end-of-group function. When the group is closed implicitly, no response is required from the controller.

**Note:** A response is always required for a system monitor session, regardless of whether the group was implicitly or explicitly closed.

### Invite Function

The invite function prepares your program to receive data. You must perform an output operation with the INVITE DDS keyword or with a system-supplied format to issue an invite function. You can combine additional output functions or data with the invite function. Your program can continue processing after issuing the invite request and does not need to wait for the data to arrive.

If a group of records was partially sent by your program, combining additional output functions or data with the invite function causes an implicit end-of-group function to be sent. If the end-of-group function is not explicitly specified (by using the end-of-group DDS keyword) with the invite function, a response to the group of records is not required.

**Note:** For a system monitor session, a response is required regardless of whether the end-of-group function was implicitly or explicitly specified.

Your program must issue a read-from-invited-program-devices operation or read operation to obtain the data.

### Read-from-Invited-Program-Devices Operation

The **read-from-invited-program-devices operation** is an input operation that waits for input from any one of the invited program devices for a user-specified time. The read-from-invited-program-devices operation follows the invite function. After issuing an invite function, use the read-from-invited-program-devices operation to receive data from any previously invited program devices. The operation always ends when any of the invited program devices have received a complete group of records, or if the time limit set by the timer function has elapsed.

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

You can use data queues for a program that waits for data on a display file, an ICF file, and a data queue at the same time (in any combination). When you specify the DTAQ parameter for the following commands:

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

you can indicate a data queue that will have entries placed on it when either of the following occurs:

- An enabled command key or Enter key is pressed from an invited display device.
- Data becomes available from an invited ICF session.

By using the IBM-supplied QSNDDTAQ program, jobs running on the system can also place entries on the same data queue as the one specified in the DTAQ parameter.

For an ICF file or display file, the application program uses the IBM-supplied QRCVDTAQ program to receive each entry placed on the data queue and then processes the entry based on
whether it was placed there by the display file, by an ICF file, or by the QSNDDTAQ program. For a display file, the application then issues a read or read-from-invited-program-devices operation to receive the data. For more information on the QRCVDTAQ function and syntax, and examples of waiting on one or more files and a data queue, see the CL Programming book.

The display file or ICF file entry that is put on the data queue is 80 characters in length and contains the field attributes described in Figure 5-2. Therefore, the data queue that is specified using the commands listed above must have a length of at least 80 characters.

Entries placed on the data queue by jobs using QSNDDTAQ are defined by the user.

**Figure 5-2. Display File and ICF File Entry Field Attributes**

<table>
<thead>
<tr>
<th>Position</th>
<th>Data Type</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 1 through 10 | Character | The type of file that placed the entry on the data queue. This field can have one of two values:  
| | | - *ICFF (ICF file)  
| | | - *DSPF (display file)  
| | | If the job receiving the data from the data queue has only one display file or one ICF file open, then this is the only field that needs to be used to determine what type of entry has been received from the data queue. |
| 11 through 12 | Binary | Unique identifier for the file. The value of the identifier is the same as the value in the open feedback area for the file. This field should be used by the program receiving the entry from the data queue only if more than one file with the same name is placing entries on the data queue. |
| 13 through 22 | Character | The name of the display or ICF file. This is the name of the file actually opened after all overrides have been processed and is the same as the file name found in the open feedback area for the file. This field should be used by the program receiving the entry from the data queue only if more than one display file or ICF file is placing entries on the data queue. |
| 23 through 32 | Character | The library where the file is located. This is the name of the library after all overrides have been processed and is the same as the library name found in the open feedback area for the file. This field should be used by the program receiving the entry from the data queue only if more than one display file or ICF file is placing entries on the data queue. |
| 33 through 42 | Character | The program device name after all overrides have been processed. This name is the same as that found in the program device definition list of the open feedback area. For file type *DSPF, this is the name of the display device where the command or Enter key was pressed. For file type *ICFF, this is the name of the program device where data is available. This field should be used by the program receiving the entry from the data queue only if the file that placed the entry on the data queue has more than one device or session invited prior to receiving the data queue entry. |
| 43 through 80 | Character | Reserved. |

**Notifying the Remote Program of Problems**

Use the fail, cancel, and negative-response functions to inform the remote program about any errors in the data being sent or received.

**Fail Function**

Use the fail function to indicate an error has occurred when sending or receiving data.

If a program that is sending data issues a fail function, this indicates that errors exist in the data sent. A cancel indication is sent to the controller program.

You can use the fail function if your program receives data and finds an error in the received data. After finding the error, your program should issue the fail function as the next operation. A negative response with a sense code of...
08110000 is sent to the controller from the AS/400 system.

**Cancel Function**

If you are sending data and decide to end a partially sent group of records, you can use the cancel function to cancel a group of records providing you have not sent the end-of-group indicator. Any buffered data is sent before the cancel request is sent.

The cancel function is not valid for the 3694 processor.

**Negative-Response Function**

If you are receiving data, use the negative-response function to inform the remote program that the data it sent contained an error.

Using a negative-response function is similar to issuing a fail function when receiving data, except that you can also include eight characters of sense data with the negative-response function. The sense data tells the sending system about the error in the data you received. The first two characters of the sense data must be either 10 or 08 or the first four characters must be 0000. All characters must be a value from 0 through 9, from A through F, or from a through f. For more information about sense data, see the Systems Network Architecture Reference Summary book.

The sense data accompanying the negative response goes to the normal output buffer. No other data can be sent with a negative-response function. If a sense code is not specified, a default sense code of 08110000 is used.

**Using Additional Functions and Operations**

The following functions give you additional control over the finance operations.

**Cancel-Invite Function**

If a program device is invited, you can use the cancel-invite function to cancel the invite function if data has not been received for the invited-program device. The cancel-invite function allows your program to cancel an invite operation, after which you can issue an input or output operation. If finance communications has already received data for the program device from the controller, the cancel-invite function is rejected with a 0412 return code.

**Timer Function**

Your program can use the timer function to set a timer and wait for it to end before performing some specified function. The timer function specifies an interval of time (in hours, minutes, and seconds) to wait before your program receives a timer-expired return code (0310).

Your program continues to run and all operations are valid during the time interval. To determine if the time limit set by the timer function elapsed, your program must use the read-from-invited-program-devices operation after issuing the timer function.

For more information about the timer function, see the ICF Programming book.

**Get-Attributes Operation**

You can use the get-attributes operation at any time after the file has been opened to determine the status of a program device. If the program device is not acquired, the information is obtained from the program device entry defined with the ADDICFDEVE or OVRICFDEVE command.

**Ending a Session**

The AS/400 application program must end the ICF finance session by using a release operation or an end-of-session function. Ending an ICF session does not necessarily affect the SNA session.

If the SNA session was established as a result of the remote system sending an INIT-SELF request, the SNA session may still be intact after your program ends the ICF finance session.
If the SNA session was established as a result of a source program issuing an ICF Acquire operation (an unsolicited start session request), the SNA session ends when the source program ends the ICF finance session.

The controller may request that the SNA session end by sending a TERM-SELF command, a Request Shutdown (RSHUTD) command, or an UNBIND command.

If no ICF session exists on the SNA session, you can end the SNA session by varying off the device using the VRYCFG command. For more information about SNA considerations, see “Systems Network Architecture (SNA)” on page 7-7.

**Release Operation**

For a source program, you can explicitly release a program device from an ICF file by using the release operation, or you can implicitly release the device by closing the file. If you release the program device, you must acquire it again before you can issue input/output operations. The release operation is accepted only if the program is not invited, if the last write operation contained an end-of-group indicator, and if no more data is waiting to be read.

For a target program, the release operation ends the logical connection between your program and the remote controller program. The program, or another program in the same job, can establish the connection again by acquiring the program device again. The communications session, including the state of the session, remains intact.

**End-of-Session Function**

The end-of-session function always ends the finance session. When your program issues an end-of-session (EOS) function, finance communications ends the program’s attachment to the session and frees the resources used during the session. Those resources are then made available to other programs waiting to establish a session. If you issue an end-of-session function when either your system or the controller program is sending or receiving data, finance communications still ends the session.

**Note:** The end-of-session function always ends the session if it exists. The only possible return codes from an end-of-session function are 0000 (normal end) or 830B (program device not acquired).

**Close Operation**

The processing done by the close operation depends on whether or not the file is shared. If the file is not shared, the file resources allocated by the open operation are deallocated and returned to the system.

If the file is shared, the program cannot do input/output operations on the file. Other programs that have the file open can still use the file.

If a close operation is successful, only an open operation is allowed to be used with the file. If the close operation fails, the program should issue the close operation a second time. A second close operation is always successful.

For a target program, the close operation ends the logical connection between your program and the remote controller program. The program or another program in the same job can establish the connection again by opening the file and acquiring the program device again. The communications session, including the state of the session, remains intact.

**Using Response Indicators**

Response indicators are specified in a 1-character field that is passed with an input record from the system to a program to provide information about the data record or actions taken by the work station user. To determine which response indicators to use when your program does a read operation, consider the following:

- What data was received from the remote program?
- What does the remote program expect of your program?
- What will be the next operation from your program?

Response indicators are only effective for input operations or a combined output, then input operation. They have no effect on an output opera-
tion. You can use multiple response indicators for a single input operation.

**Receive End-of-Group**

Use the receive end-of-group (RCVENDGRP) response indicator to determine if your program received a group of records. You receive one end-of-group record only once per group, and only one record contains the end-of-group indication. The major return code can be either 00, 02, or 03 and the minor return code is 03 or 07.

**Receive Function-Management-Header**

Use the receive function-management-header (RCVFMH) indicator to determine if your program received a function-management-header indicator. The major return code is 00 and the minor return code is 07 for the function-management-header indicator.

**Receive Negative-Response**

Use the receive negative-response (RCVNEGRSP) indicator to determine if an error exists in data that your program has sent. The major return code is 83 and the minor return code is 19 for the data sent by the RCVNEGRSP indicator.

**Using Input/Output Feedback Area**

The input/output (I/O) feedback area contains the results of read and write operations for your finance application programs as communicated by ICF return codes. This feedback area consists of two parts:

- A common input/output feedback area containing information affecting all communications types
- A file-dependent input/output feedback area containing information affecting one or more communications types

Offset 38, in the file-dependent part of the I/O feedback area, applies to finance. This field contains negative-response error data. For return code 8319, this area contains sense data indicating the reason the operation failed. For more information about sense codes, see the book SNA Formats.

For more information about the input/output feedback area, see the ICF Programming book.

**Using Return Codes**

After an operation ends, a return code (and a high-level language file status) is sent to your application. This return code indicates if your operation ended successfully. Exception messages can also be issued along with the return code. For more information about return codes, see Appendix B, “Return Codes, Messages, and Sense Codes.” For more information about the high-level file status, see the appropriate language reference book.

**Note:** ILE C/400 language does not use high-level file status. However, your ILE C/400 program can use the errno.h statement to check for any I/O exceptions that may occur and to retrieve any exception data associated with the error. Your program should check the return code to determine the following:

- Status of operation just ended
- Operation to be done next

For example, when issuing a read operation, a major return code of 00 on an input operation indicates that the operation ended successfully. Along with this major code, you can also receive one of the following minor return codes from finance communications:

- **01** Indicates your program completed a successful invite function.
- **03** Indicates that your program has received a group of records.
- **07** Indicates that a function-management-header was received by your program in this group of records.

Another example is a major code of 83, which indicates an error was found that could be recoverable. Different minor codes can be returned with this major code, just as with major return code 00. For example, if your program receives a return code of 831C, this means your program failed to correctly process a return code to a previous write operation. Data was available to be
received. To correct the problem, issue an input operation to receive the data.

Your program should check ICF return codes when every operation ends to be certain that the operation completes successfully or that appropriate recovery action was taken.

For a description of return codes that can be returned to your finance application, see Appendix B, “Return Codes, Messages, and Sense Codes.”
Chapter 6. Writing Non-Intersystem Communications Function Application Programs

The information in this chapter is intended to help the customer write finance application programs using the non-ICF interface. It contains general-use programming interfaces, which allow the customer to write programs that use the services of the OS/400 program.

This chapter describes starting a non-ICF finance application program using the Submit Finance Job (SBMFNCJOB) command. It also discusses using finance input/output manager (FIOM) and user-defined data stream (UDDS).

Using non-ICF finance represents three levels of support, with the SBMFNCJOB command providing the most support. Your application programs are not required to do any communications operations, since non-ICF finance communicates through a display file using UDDS. UDDS gives your program total control of the data stream. Using FIOM allows your program to communicate with finance devices by calling input/output (I/O) routines rather than doing communications operations.

Figure 6-1 shows the input data format for non-ICF finance. The incoming data contains a 3-byte header. The outgoing data format for non-ICF finance must contain UDDS control bytes and a 4700 finance control byte, as shown in Figure 6-2.

Using the Submit Finance Job Command

This section describes the SBMFNCJOB command, including how and when to use it. Processing examples and information describing how to set up a device, user, and program tables are
included. These tables are used by the SBMFNCJOB command. Error handling is also discussed. For more information and a syntax diagram of the SBMFNCJOB command, see the CL Reference book.

A finance job started with the SBMFNCJOB command is an interface between the AS/400 finance application programs and the 4701 or 4702 controller application programs.

Use the SBMFNCJOB command if all of the following situations exist:

- The AS/400 system communicates with a 4701 or 4702 finance controller.
- A finance device table and a finance program table have been defined (defining a finance user table is optional) using the WRKDEVTBL and WRKPGMTBL commands as described in “Work with Device Table Command” on page 3-5 and in “Work with Program Table Command” on page 3-7.
- The device types 3277, 3278, or 3279 are not entered in your device table.
- The USER parameter supplied on the job description, under which the finance job runs, is USER (QFNC). Specify this parameter using either the Create Job Description (CRTJOBD) or Change Job Description (CHGJOBD) command. For more information about these commands, see the CL Reference book.
- The 4701 or 4702 controller application program sends data, or transactions, first and expects to receive data back.
- The 4701 or 4702 controller application program passes data in the proper format, as described in “Processing Transactions” on page 6-6.

The SBMFNCJOB command submits a batch job to the QFNC subsystem through the QFNC job queue. This batch job does the following:

- Acquires the devices in the device table.
- Invites these devices to allow data to be received from them.
- Verifies that the user ID received with the INIT-SELF request matches an entry in the finance user table. This applies only if a user table was created and its name was specified in the SBMFNCJOB command.
- Calls the program requested by the finance controller to process the transaction if the program is specified in the program table.
- Returns data formatted by your AS/400 application to the finance controller.
- Releases the device when your finance controller requests a session end.

**Parameters**

When using the SBMFNCJOB command, specify the following parameters:

**DEVTBL**

Specifies the name of the device table that the finance job uses to determine which 4704 or 3624 devices the finance job controls. This is a required parameter.

**PGMTBL**

Specifies the name of the program table the finance job uses to determine, from the program ID (sent in the data stream with a finance transaction), which system user program names process the finance transaction. This is a required parameter.

**USRTBL**

Specifies the name of the user table that the finance job uses to verify a finance user when a finance sign-on is received.

**JOB NAME**

Specifies the job name that is associated with the submitted finance job. The possible values are:

- **QFNCJOB**: The job is submitted as QFNCJOB.
- **Job_name**: Specify the user-defined job name that is associated with the submitted finance job.

**JOBD**

Specifies the name of the job description the finance job uses.

**MSGQ**

Specifies the name of the message queue, if any, to which messages are sent while the finance job is running.
**Error Handling**

This section describes error handling support for a finance job started with the SBMFNCJOB command.

**Input/Output:** Finance support attempts error recovery whenever a finance job receives an I/O exception. When an I/O exception signals a finance job, the major/minor return code is retrieved from the message to determine the potential of recovery from the error. Recoverable errors alert the finance job to try recovery. If successful, processing continues normally. If unsuccessful, further action depends on the nature of the error.

Device errors result in the release of the affected device and continued processing of other devices associated with the job. However, if recovery is not successful, the controller or line errors end the job.

When a permanent I/O error is received, that error is handled the same as an unsuccessful recovery. Device failures cause the release of the affected device but communication with remaining devices is continued. If other I/O exceptions are too severe to be handled within the job, the job ends.

**Non-Input/Output:** For a finance job, the SBMFNCJOB command handles non-I/O errors as most AS/400 jobs are handled. If an error occurs, a message is sent to the message queue associated with the finance job. The SBMFNCJOB command also allows you to specify the name of the message queue where you want certain informational messages sent. Messages sent to this finance queue relate to general error conditions that happen when the finance job runs.

Some of the informational messages include:
- CPI8372 - ERRORS OCCURRED DURING THE START OF FINANCE JOB
- CPI8379 - FINANCE TRANSACTION FAILED
- CPI8380 - FINANCE DEVICE FAILED
- CPI8394 - PROGRAM ID NOT IN PROGRAM TABLE

This additional message queue allows you to supervise the status of your finance jobs more effectively.

For additional information about AS/400 finance support messages, see the online message help information.

**Supervising Finance Jobs**

To supervise finance jobs, you can do the following:

- Specify MSGQ (*NONE) on the SBMFNCJOB command when you submit a job. Then, if an error occurs when starting finance support, a message appears in the message queue.
- Use the Work with Configuration Status (WRKCFGSTS) command to ensure that all devices allocated to a finance job have been successfully acquired.
- Use the Work with Job Queue (WRKJOBQ) command to display any finance jobs that have been submitted and are waiting to run when the QFNC finance controller is inactive:
  ```
  WRKJOBQ JOBQ(QGPL/QFNC)
  ```
- Use the Work with Active Job (WRKACTJOB) command to display and change the status and performance information for submitted finance jobs:
  ```
  WRKACTJOB SBS(QFNC)
  ```
- Use the Display Job Log (DSPJOBLOG) command, with the job names as parameters, to display messages held in the job log when running your finance jobs.

For an explanation of the format and presentation of the IBM-supplied messages and the complete finance support message texts, see the online messages.

For more information about these commands, see the CL Reference book and Work Management book.

The SBMFNCJOB command has a JOB parameter that makes supervising and canceling finance jobs easier. Use this parameter to supply unique names for your finance jobs so that you can easily distinguish them from other jobs in your system.
Data Flow Examples

The following figures show data flow examples when using the SBMFNCJOB command.

Figure 6-4 on page 6-4 shows a finance job verifying that the user ID received from the controller application exists in the user table associated with that job.

Figure 6-4 shows how a user ID sent in the INIT-SELF data stream is validated using the user table named on the SBMFNCJOB command parameter.

Figure 6-5 on page 6-5 represents an environment in which more than one finance job is submitted. The figure shows that two finance jobs control the finance devices. The finance jobs share certain application programs but cannot share devices.

Notice in Figure 6-5 that DEVICEA is acquired by FNCJOB1 and devices DEVICEB and DEVICEC are acquired by FNCJOB2. The two finance jobs cannot share devices. Even if only one device name is specified in two device tables, only one finance job can acquire that device. However, finance jobs can share application programs. The two finance jobs share programs PGMA and PGMD in Figure 6-5 on page 6-5.
Figure 6-5. Two Finance Jobs Control the Finance Devices
Figure 6-6 shows how a transaction sent in a data stream from the controller application, with the finance job calling the requested transaction processing program, can be handled.

**Processing Transactions**

By using the SBMFNCJOB command, financial transactions can be processed from the 4701 or 4702 controller on the AS/400 system, or from the AS/400 system on the 4701 or 4702 controller. This topic describes how each process occurs.

**Sending Data from the Finance Controller to the AS/400 System:** When you use the SBMFNCJOB command, the finance controller application program must use one of the data stream formats shown in Figure 6-7.

Figure 6-7 shows two data streams being sent from the finance controller. These data streams represent the two formats accepted by the finance job. The first data stream format contains the characters *EXEC, followed by a single blank, a program ID, another blank, and data. The second format has no *EXEC characters; this data stream begins with the program ID, which is again followed by a single blank and data.

When you use either data format, the program ID (PGM ID) can be of variable length from 1 to 8 characters and must be followed by a single blank. A maximum of 512 bytes of data is allowed per transaction.

**Figure 6-6. Controller Processing Transaction T001**

In Figure 6-6, the controller requests processing of transaction T001. The SBMFNCJOB interface determines from the program table that the AS/400 system application PGMA should process the transaction.
When the finance job receives the data stream from the finance controller, it uses the finance program table you created to determine which of your AS/400 application programs to call. Your AS/400 application program is passed through the following SBMFNCJOB parameters:

**device name**
A character variable, 10 bytes in length, representing the name of the 3624 or 4704 device sending the data.

**data length**
A decimal field, 15 bytes in length with 5 decimal positions, containing the length of the data received.

**data**
A character variable, 512 bytes in length, containing the data received from the finance controller. The data length parameter determines the actual length of the finance data in the data parameter.

Figure 6-8 shows the expected parameters.

![Figure 6-8. Parameters Expected by the AS/400 Application Program](RV2P866-0)

**Note:** The maximum size of a character variable (referred to in Figure 6-8) might be restricted to fewer than 512 bytes by the high-level language in which your application programs are written. If so, see the programmer’s guide for the programming language you are using, or the user’s guide for the controller application to ensure that the controller does not send more data than your AS/400 application program can receive.

**Sending Data from the AS/400 System to the Finance Controller:** To send data from your AS/400 application program back to the finance controller application program:

1. Set the data length parameter to the length of the data you want to send.
2. Move your new data into the data parameter.

If you do not want to send data back to the finance controller application program, set the data length parameter to 0 bytes.

**Using Finance Input/Output Manager**

This section describes the finance input/output manager (FIOM) call, including how and when you can use it. Processing specifications and error handling are also discussed.

FIOM is a flexible tool for use in a variety of finance environments. FIOM supplies a high-level connection to UDDS communications for AS/400 finance support users. It simplifies logical unit 0 (LU0) communications between your AS/400 transaction application program and the controller application program. FIOM is an alternative to transparent UDDS communications when you want direct conversation between your AS/400 transaction-processing application program and the finance controller application.

The FIOM support formats, sends, and receives UDDS. Figure 6-9 and Figure 6-10 on page 6-8 compare communications using UDDS and FIOM. In transparent UDDS communications, shown in Figure 6-9, your AS/400 application program communicates with the controller application using SNA LU0 protocols. Data streams must be framed in UDDS control information.

![Figure 6-9. Transparent UDDS Communications](RV2P867-0)

FIOM makes this communication easier as shown in Figure 6-10 on page 6-8. Information you supply to the FIOM routines allows for UDDS communication, yet removes many of the requirements for formatting and sending UDDS in your high-level application.
Finance Input/Output Manager Routines

FIOM is called by your program. FIOM consists of four routines that can be called as external subroutines by your application programs to do I/O operations on a finance device that is configured as a non-ICF device. Each of the four routines accepts the parameters you supply and then performs the input/output (I/O) functions.

The following discussions supply a description of each routine. Each of the four FIOM routines discussed has an associated DATA parameter.

The DATA parameter for the following four routines allows a 512-byte field. However, an ILE RPG/400 program sets the maximum length of a character field at a value of 256 bytes. To determine the maximum length of data that can actually be sent for your application programs, see the appropriate programmer's guide.

**QFN-Write Routine:** The QFN-write (QFNWRT) routine accepts data from your AS/400 application program and sends the formatted data to the finance device you specify.

The following formats for the input parameters are passed to the QFNWRT routine:

- **Device ID**
  A 10-byte character field specifying the name of the finance device to receive the data.

- **Data length**
  A decimal field containing the data length to be written. The data length is defined as 15 bytes with 5 decimal positions. The maximum data length is 512 bytes.

- **Data**
  A character field containing data to be written to the device. The format of the data depends on the value assigned to the data type parameter. If the value of the data type is 0, the format of the data depends on protocols established for communications between your AS/400 application and the controller application program. If the value of the data type is 1, QFNWRT ignores the data parameter because the data stream being sent to the controller application has a specific, predetermined format. Other data type values, 2 and 6, correspond to data stream formats that you define to satisfy both SNA and AS/400 application program requirements.

**Data type**
- A 1-byte character field that contains the type of data (4700 control character) written to the finance device. Valid data types are:
  - 0: 3694, 4701, and 4702
  - 1: Positive response
  - 2: Negative response, with the data parameter containing a minimum of 2 bytes of status code followed by sense data
  - 6: Send 3694 data type with function-management-header

The following example is a call to the QFNWRT routine from an ILE RPG/400 program:

```
CALL 'QFNWRT'
PARM WSID "1/zerodot"
PARM SNDLEN 155
PARM DATA 256
PARM WRTTYP 1
```

**QFN-Write/Invite Routine:** The QFN-write/invite (QFNWRTI) routine works in the same manner as the QFNWRT routine, and also invites a response from the finance device.

The following formats for the input parameters are passed to the QFNWRTI routine:

- **Device ID**
  A 10-byte character field specifying the name of the finance device to receive the data and to be invited for communication.

**Data length**
A decimal field containing the length of the data to be written. The data length is defined as 15 bytes with 5 decimal positions. The maximum data length is 512 bytes.

**Data**
A character field containing data to be written to the device. The format of this data...
depends on the value assigned to the data type parameter. If the value of the data type is 0, the format of the data depends on protocols established for communications between your AS/400 application and the controller application program. If the value of the data type is 4 or 5, the data parameter is ignored by the QFNWRTI routine because the data stream being sent to the controller application has a specific, predetermined format. Other data type values, 3 and 6, correspond to data stream formats that you define to satisfy both SNA and AS/400 application program requirements.

Data type
A 1-byte character field containing the type of data (4700 control character) written to the finance device. Valid data types are:

0 3694, 4701, and 4702
3 LUSTAT command follows, with the data parameter containing the LUSTAT X'04' command, followed by a minimum of 2 bytes of status
4 Request INIT-SELF command from finance device
5 Invite the finance device without writing data to the device
6 Send 3694 data type with the function-management-header

This is an example of a call to a QFNWRTI routine from an ILE RPG/400 program:

CALL 'QFNWRTI'
PARM WSID 1/zerodot
PARM SNDLEN 155
PARM DATA 256
PARM WRTTYP 1

QFN-Read Routine: The QFN-read (QFNREAD) routine accepts data sent from a specific finance display in response to an invite operation sent by the QFNWRTI routine.

The following parameter formats are passed to the QFNREAD routine:

Device ID
A 10-byte character input field specifying the name of the finance device from which to read.

Data length
A decimal output field containing the length of the data read. The data length is defined as 15 bytes with 5 decimal positions. The maximum data length is 512 bytes.

Note: The data length parameter must be started by your application program and must contain a valid packed decimal (15 5) number before the QFNREAD routine is called.

Data
A character output field containing the data read from the finance device. The format of this data depends on the parameter value of the data type. If the value of the data type is 1, the format of the data depends on protocols established for communications between your AS/400 application and the controller application program. If the value of data type is 3, the data stream returned to the AS/400 application has a specific, predetermined format.

Note: The data parameter must represent a field in your application program that is large enough to contain the expected input data. If the receiving field is not large enough, adjacent data space can be overwritten with financial data.

Data type
A 1-byte output character field containing the type of data (4700 control character) read from the finance device. Valid data types are:

1 3694, 4701, and 4702 (no function-management-header)
3 Function-management-header and data to follow (INIT-SELF, TERM-SELF, or 3694, 4701, and 4702 data)

This is an example of a call to a QFNREAD routine from an ILE RPG/400 program:

CALL 'QFNREAD'
PARM WSID 1/zerodot
PARM RCVLEN 155
PARM DATA 256
PARM RDTYP 1
QFN-Read/Invited Routine: The QFN-read/invited (QFNREADI) routine accepts input from any one of the invited finance devices associated with the finance job, returning the data to your application along with the name of the device from which it was received.

The following parameter formats are passed to the QFNREADI routine:

**Device ID**
A 10-byte output character field that specifies the name of the finance device from which data was read.

**Data length**
A decimal output field containing the length of the data that was read. The data length is defined as 15 bytes with 5 decimal positions. The maximum data length is 512 bytes.

*Note:* The data length parameter must be started by your application program and must contain a valid packed decimal (15 5) number before QFNREADI is called.

**Data**
A character output field containing the data read from the finance device. The format of this data depends on the value of the data type parameter. If the value of the data type is 1, the format of the data depends on protocols established for communications between your AS/400 application and the controller application program. If the value of the data type is 3, the data stream returned to the AS/400 application has a specific, predetermined format.

*Note:* The data parameter must represent a field in your application program that is large enough to contain the expected input data. If the receiving field is not large enough, adjacent data space can be overwritten with financial data.

**Data type**
A 1-byte character output field that contains the type of data (4700 control character) read from the finance device. Valid data types are:

1. 3694, 4701, and 4702 data
2. Function-management-header and data to follow (INIT-SELF, TERM-SELF, or 3694, 4701, and 4702 data)

This is an example of a call to a QFNREADI routine from an RPG/400 program:

```
CALL 'QFNREADI'
PARM WSID 10
PARM RCVLEN 155
PARM DATA 256
PARM RDTYP 1
```

**Error Handling**
Any errors received by the FIOM routines during transaction processing produce diagnostic messages describing the errors. The CPF8390 escape message appears after these messages. For example, if the FIOM routines receive an I/O error, the diagnostic message CPD8289 (I/O errors occurred) is signaled and the escape message CPF8390 (errors occurred when running the program) is sent. Figure 6-11 on page 6-11 shows a list of the message identifiers sent by the FIOM routines.

The ability of your application program to handle error conditions depends on the language in which the program is written. For specific error-handling capabilities and error-recovery procedures for ICF finance, see the programmer's guide for the language.
If the programming language has error-handling capabilities, the application program can attempt recovery after receiving an I/O error from FIOM. I/O, OPEN, and CLOSE messages supply the major and minor return codes of the operation as part of the message replacement text. If the application program can retrieve this information, then you can try error recovery.

If the AS/400 finance job receives I/O errors during processing, and if the finance controller indicates host system format errors, examine the format of data streams being sent by the system application. These exceptions usually indicate that the data is not formatted correctly.

### Using Finance Input/Output Manager with the Submit Finance Job Command

FIOM can extend the communications capabilities of your AS/400 finance application programs when the finance job is used as the primary interface to the finance controller application.

As discussed in “Processing Transactions” on page 6-6, when a finance job receives a transaction, one of your application programs is called to process the transaction. With regard to that application, you can expect the following to occur to a finance job submitted through the SBMFNCJOB command:

- Your application receives the finance device name, data length, and data as input parameters.
- Your application returns the data length and data parameters as output.
- When control returns to the finance job, the device named as input to your application remains in the same status as it was prior to the start of your program. Specifically, the following must occur:
  - The device must be acquired.
  - The device must have an active session. Therefore, if a TERM-SELF request is received by your application program, you must either send a negative response or process the TERM-SELF request and not return control to the finance job until another INIT-SELF request is received, and a session is active again with the device.
  - The device must be ready to be invited for further communications. The finance device is invited when the data you return from your program is written to that device.
- When control returns to the finance job, all devices, except the device named as input to your application, exist (acquired or unacquired; invited or uninvited; session active or inactive) as they did before your application program was called. For example, assume...
that within your application, the QFNREADI routine is called to receive data from any invited finance device. After data is received from a device, call the QFNWRTI routine to invite the device again. In this way, when control returns to the SBMFNCJOB interface, the device is invited as it was before your application program was called.

If the preceding conditions do not occur, when control returns to the finance job, the results cannot be predicted. If the conditions do occur, the SBMFNCJOB interface allows independent communications between your application program and the finance controller application. After a finance job starts your program, and if the system exists as described in the previous paragraphs, your program can communicate directly with the controller application when control is returned to the finance job. Figure 6-12 on page 6-13 is a diagram of this communication.

Figure 6-12 on page 6-13 shows an environment in which a finance job is active on the AS/400 system using the OS/400 licensed program. When the finance job receives a transaction, one of your application programs is called to process the transaction. In turn, your program can call the four FIOM routines to communicate with the device. When the communications and all other transaction processing completes, your program should return control to the finance job, keeping the conditions previously described. The finance job returns the data supplied as output from your application program to the finance device sending the transaction. The device is invited again, and the finance job is ready to receive another transaction.
An advantage gained from using FIOM in a finance job environment is having more than 512 bytes of data associated with a particular transaction. If, for example, your ILE RPG/400 application program must write 800 bytes of data to complete a transaction, the data stream can be made into segments of 256, 256, 256, and 32 bytes. The 256-byte segments can be returned to the device through calls to the QFNWRT routine, with the last 32 bytes of data returned through the finance job. While this capability exists through UDDS, using FIOM is an easier method.

Using Finance Input/Output Manager without the Submit Finance Job Command

You can use FIOM without the SBMFNCJOB routine to simplify communications between your application programs and the controller application program. In environments where communications is routed by another method other than the SBMFNCJOB command, the four FIOM routines simplify communications for the routing program itself.

Figure 6-13 on page 6-14 shows how to use FIOM to handle communications for both the communications router and the application programs.
Your Application Programs and Communications Router

This diagram shows an environment in which the finance job is not used on the AS/400 system. Your programs are responsible for routing the finance communications and for processing all data streams received from the finance controller. Since your programs are communicating directly with the controller, use the FIOM routines to simplify this communications by handling the read, write, and invite operations when called by your AS/400 application.

Your program is also responsible for acquiring and releasing the devices. To handle UDDS, you must compile your program against an externally described display file named QDFNDATA. This display file contains the following record formats:

- `R UDDS1 KEEP
  DATA 518 B 1 2`
- `R UDDS2 INVITE
  DATA 518 B 1 2`

This file must be placed before the QSYS library in your library list. After compiling the program, remove the library containing the file from the library list. Then your application program uses QDFNDATA file in QSYS library.

Notice in Figure 6-13 that the direct communications path still exists between the router and controller applications. This indicates additional communications occurrences, such as your router opening and closing the QDFNDATA display file or acquiring and releasing finance devices. While these tasks must be done independently of FIOM, a similar principle can be used to handle the communications; you can write your own subroutines to open and close the file and to acquire and release devices.

### Using User-Defined Data Stream

This section discusses formatting information for UDDS and contains examples of communicating using UDDS.

Rather than using finance support, you can use UDDS to control and process the data streams. You must define a display file with record formats containing the user-defined (USRDFN) keyword. Then you can perform the usual input and output operations on the device by using these record formats.

Specify the USRDFN keyword at a record level (fields are not allowed on formats) by using the following steps.

1. Define an externally described display file and create your program using the record formats in this file. The file must not have record formats with the USRDFN keyword. Note the following example:

   - `R UDDS1 KEEP
     DATA 518 B 1 2`
   - `R UDDS2 INVITE
     DATA 518 B 1 2`

2. Create a second file with the same name as the first file. When you create this file, you must specify LVLCHK("NO) in the Create Display (CRTDSPF) command.
The record format in this file must contain the USRDFN keyword. Note the following example:

```
R UDDS1 USRDFN
KEEP
R UDDS2 USRDFN
INVITE
R DATAREC
DATA 518 1 2
```

3. When running your program, use the second display file.

The INIT-SELF and TERM-SELF requests are sent to the AS/400 system by the controller. The AS/400 application program must do the following:

- Respond to the INIT-SELF and TERM-SELF commands.
- Process the transaction requested by the finance terminal operator.
- Send a write instruction to communicate with the finance terminals.
- Process the data stream associated with the write instruction in the UDDS format.

Formats

This section provides examples of the format that UDDS control bytes must follow.

Control Bytes: The finance control bytes for UDDS are shown in Figure 6-14 and Figure 6-15.

```
<table>
<thead>
<tr>
<th>UDDS Control Bytes</th>
<th>Finance Control Character</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BYTE</td>
<td>CHARACTER VALUE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 - Send SNA positive response to INIT-SELF or TERM-SELF</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 - Send SNA negative response to INIT-SELF or TERM-SELF</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2 - Send SNA negative response to INIT-SELF or TERM-SELF, where the data contains 4 bytes of SNA sense data</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3 - LUSTAT command follows, where the data contains the LUSTAT '04'X command followed by 4 status bytes</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4 - Request INIT-SELF from finance device</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 - Invite data without writing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 - Send data with function-management-header (FMH) (used for 3694 controller)</td>
<td></td>
</tr>
</tbody>
</table>
```

**Figure 6-14. UDDS Control Bytes**

```
<table>
<thead>
<tr>
<th>UDDS Control Bytes</th>
<th>Finance Control Character</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BYTE</td>
<td>CHARACTER VALUE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 - No FMH included in data</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 - FMH precedes actual data</td>
<td></td>
</tr>
</tbody>
</table>
```

**Figure 6-15. UDDS Finance Control Character**

Data coming from the 3694, 4701, or 4702 controller is shown in Figure 6-16. The data field follows three bytes of control information.

```
<table>
<thead>
<tr>
<th>Reserved</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - No FMH included in data</td>
<td></td>
</tr>
<tr>
<td>3 - FMH precedes actual data</td>
<td></td>
</tr>
</tbody>
</table>
```

**Figure 6-16. Format of Data from a 3694, 4701, or 4702 Controller**

The AS/400 application program you use to communicate with a controller (using UDDS) must do the following:

- Acquire the finance devices.
- Identify and respond to the INIT-SELF requests.

The application program issues a write/read operation using UDDS to receive the INIT-SELF request.

- Set the 4700 control character.
- Verify the finance terminal operator IDs and restrict their use, if necessary.

When you receive an INIT-SELF request, your application program must send either an SNA positive or negative response and the correct 4700 control character.

- Call your transaction processing program.

After your AS/400 application program sends the positive or negative response to the INIT-SELF, the application program can either write data to the controller (4700 control character set to 0), or it can invite the controller to...
send data by issuing a write/read with the 4700 control character set to 5.

- Release the finance device when you receive a TERM-SELF request.

When your AS/400 application program receives the TERM-SELF, it must issue a write operation to send a positive response (4700 control character set to 1) and then release the device, or send a negative response (4700 control character set to a value of 2 bytes) and continue communications with the device.

**Data Streams**

The following sections describe the format of data streams that are sent and received by your AS/400 application programs when you use UDDS.

**INIT-SELF Request:** The format for the INIT-SELF request is:

X'00020373F4'.

**Bytes 1 and 2 (X'0002')**
Length of data passed from the AS/400 application to the finance controller application beginning in byte position 6 of the UDDS.

The minimum length of any UDDS is 2 bytes. Setting byte position 6 to X'F4' causes the extra byte at the end of the data stream to be ignored.

**Bytes 3 and 4 (X'0203')**
Length of data received from the finance application program. This length is set to X'0203' (decimal 515). It accepts a maximum of 512 bytes of data plus the three control characters before it.

**Byte 5 (X'73')**
UDDS control flag. It is set to a value representing a write/read operation.

**Byte 6 (X'F4')**
4700 control character. The 4-character value is an INIT-SELF request from the finance controller application program.

**INIT-SELF Data Stream:** The INIT-SELF data stream that the AS/400 application program receives from the finance controller application program is:

X'0101F301068100...mmmmF308xxxx...xxx0000nnnnuuuu...uuu'.

**Bytes 1 and 2 (X'0101')**
Reserved bytes.

**Byte 3 (X'F3')**
Formatted data will be present in the data stream.

**Bytes 4 through 6 (X'010681')**
SNA command for INIT-SELF request.

**Byte 7 (X'00')**
Reserved/control information.

**Bytes 8 through 15 (represented: mmmm...mmmm)**
Mode.

**Bytes 16 through 25**
Name of the destination logic unit (DLU).

**Byte 16 (X'F3')**
Type of logical unit.

**Byte 17 (X'08')**
Length of the symbolic name.

**Bytes 18 through 25 (represented: xxxx...xxxx)**
Symbolic name as either c'DTNCHXVS' for the 3694 processor, or c'SFSbbbb' for the 4701 controller.

**Byte 26 (X'00')**
Requester ID length, no requester ID.

**Byte 27 (X'00')**
Password length, no password.

**Bytes 28 through 50**
User field.

**Byte 28 (represented: nn)**
Length of the user data.

**Bytes 29 through 50 (represented: uuuu...uuuu)**
User data.

The user field could contain the sign-on to the finance device. Your AS/400 application program should verify that the user ID is valid when the SBMFNCJOB command is not used. To do this, your AS/400 application can use a table of valid
user IDs to approve the user ID passed in the user field of the INIT-SELF data stream. See “Work with User Table Command” on page 3-6 for information about user ID tables. (You may start the Display File Field Description (DSPPFD) command on the QUSRSYS/QFNUSRTBL file to determine the attributes of the user table file.) If the ID is not valid, the program may request the correct user ID.

The finance job assumes that the first 8 characters of the user data field contain the user ID from the controller application program.

The INIT-SELF request can pass 22 bytes of data from the controller application to the AS/400 application.

**Positive Response to INIT-SELF Request:** The format for a positive response to INIT-SELF request is:

\[ X'00002000071F1' \]

**Hexadecimal data value**

\[ 0002+0000+71+F1 \]

**Byte position**

\[ 1 2 3 4 5 6 \]

### Bytes 1 and 2 (X'0002')
Length of data passed from the program to the finance controller application beginning in byte 6 of the UDDS.

The minimum length of any UDDS is set at a value of 2. Setting byte position 6 to X'F1' causes the extra byte at the end of the data stream to be ignored.

### Bytes 3 and 4 (X'0000')
Length of data received from the finance application program. The length is set to 0 bytes to indicate that no data will be received from the controller.

### Byte 5 (X'71')
UDDS control flag. It is set to a value representing a write only operation.

### Byte 6 (X'F2')
4700 control character. The character 2 value indicates a negative response to the data received from the finance controller application program.

### Bytes 7 through 10 (X'08xx0000')
Returned to the controller as a negative response. The xx response code can be replaced with values, such as X'0F'—Not authorized to session or,X'35'—Invalid parameter (invalid data length).


### Bytes 11 through 13 (X'010681')
Returned to the controller application with bytes 7 through 10. This indicates that the negative response refers to an INIT-SELF data stream format.

**TERM-SELF Data Stream:** The TERM-SELF data stream that the AS/400 application program receives from the controller application program is:

\[ X'0008000071F208xx0000010681' \]

**Hexadecimal data value**

\[ 0008+0000+71+F2+08xx0000+010681 \]

**Byte position**

\[ 1 2 3 4 5 6 7 10 11 13 \]

### Bytes 1 and 2 (X'0008')
Length of data passed from the program to the finance controller application beginning in byte 6 of the UDDS.

### Bytes 3 and 4 (X'0000')
Length of data received from the finance application program. This length is set to 0 bytes to indicate that no data will be received from the controller.

### Byte 5 (X'71')
UDDS control flag. It is set to a value representing a write only operation.

### Byte 6 (X'F3')
Formatted data follows in the data stream.
Bytes 4 through 6 (X’010683’)
SNA command for TERM-SELF.

Byte 7 (X’00’)
Reserved/control information.

Bytes 8 through 17
Name of the destination logical unit (DLU).

Byte 8 (X’F3’)
Type of logical unit.

Byte 9 (X’08’)
Length of the symbolic name.

Bytes 10 through 17 (represented: xxxx....xxxx)
Symbolic name either as c’DTNCHXVS’ for the 3694 controller, or c’SFSbbbb’ for the 4701 controller.

Positive Response to TERM-SELF Request:
The format for the positive response to TERM-SELF request is:
X’/zerodot/zerodot/zerodot2/zerodot/zerodot/zerodot/zerodot71F1’.

Hexadecimal data value 0002+0000+71+F1
---+---+---+---
Byte position 1 2 3 4 5 6

Bytes 1 and 2 (X’0002’)
Length of data passed from the program to the finance controller application beginning in byte 6 of the UDDS.

Bytes 3 and 4 (X’0000’)
Length of data received from the finance controller application program. This length is set to 0 bytes to indicate that no data will be received from the controller.

Byte 5 (X’71’)
UDDS control flag. It is set to a value representing a write only operation.

Byte 6 (X’F2’)
4700 control character. The character 2 value indicates a negative response to the data received from the finance controller application program.

Bytes 7 through 10 (X’08xx0000’)
Returned to the controller as a negative response. The xx response code can be replaced with a value, such as X’16’—Session does not exist.


Bytes 11 through 13 (X’010683’)
Returned to the controller application with bytes 7 through 10 to indicate that the negative response refers to an INIT-SELF data stream format.

Logical Unit Status Command:
The Logical Unit Status (LUSTAT) command can be used by your program to report failures in the finance controller application. The format for the LUSTAT data stream is:

Hexadecimal data value 0006+0000+73+F3+/zerodot/zerodot/zerodot/zerodot+uuuu
---+---+---+---+---+---+---+---
Byte position 1 2 3 4 5 6 7 8 9 10

Bytes 1 and 2 (X’0006’)
Length of data passed from the program to the finance controller application beginning in byte 6 of the UDDS.
Bytes 3 and 4 (X'0203')

Length of data received from the finance controller application program. This length is set to 515 bytes to indicate that a maximum of 512 bytes of data and 3 bytes of control information is received from the controller.

Byte 5 (X'73')

UDDS control flag. It is set to a value representing a write/read operation.

Byte 6 (X'F3')

4700 control character. The character 3 value indicates that an LU status message is being sent to the finance controller application program.

Bytes 7 (X'04')

LUSTAT request code.

Bytes 8 and 9 (X'0000')

Status value for the LUSTAT data stream. These bytes are set to zero to indicate that the user status follows.

Bytes 10 and 11 (X'uuuu')

Status extension field. This file defines the exact message being sent to the finance controller. The values for this field include:

- X'0000': Session does not exist
- X'0001': Program ended normally
- X'0002': Program ended abnormally
- X'0003': Resource now available
- X'0004': Disable pending
- X'10nn': Procedure start failed, where nn indicates one of the following:
  - 00: No additional information
  - 01: Sign-on has invalid library name
  - 02: Disk I/O error in security record
  - 03: Job's starting ended by system operator
  - 04: Resources not available to start job
  - 05: Resource security file not found
  - 06: Cannot log security information to history file
  - 07: No user list in resource security file for library
  - 08: Unauthorized request for user library

A finance job started with the SBMFNCJOB command uses LUSTAT data streams with the user status fields of X'0002' and X'1009'. The X'0002' message is sent when a request for an application program by the program ID (included with the transaction) fails. The X'1009' message is sent when the program ID included with the transaction does not exist in the program table associated with the finance job.


3694 Communications with User-Defined Data Stream

Communication between the AS/400 application program and the CHX/3694 program is controlled by function-management-headers. For more information about function-management-headers, see the Check Processing Executive/VS: Program Logic Manual. For more information about programming for the 3694 controller, see the Check Processing Executive/3694: Program Reference and Operations Manual and the Check Processing Executive/VS: Program Reference and Operations Manual.

Function-Management-Headers: A 3694 function-management-header is a special record (or portion of a record) that contains control information for the data that follows. The first byte is the length of the header. The length is in hexadecimal values and includes the length byte. The header portion immediately follows the length byte.

The 3694 processor verifies the data before the function-management-header is sent to the AS/400 system. When the AS/400 system receives the data, it identifies the header and sets the data type byte to C'3' to indicate to the program that the header was received.

Input Data Format: The data (read by the program) has the following format when a function-management-header is received from the 3694 processor:
Bytes 1-2
  Reserved

Byte 3
  '3'

Byte 4
  Function-management-header length (X'02')

Byte 5
  Function-management-header identifier (X'80')

Bytes 6-7
  Function-management-header type

Bytes 8-n
  Application data

The data (read by the program) has the following format when a function-management-header is not received from the 3694 processor:

Bytes 1-2
  Reserved

Byte 3
  '1'

Bytes 4-n
  Application data

Output Data Format: The data written by your AS/400 application program must have this format if the data contains a function-management-header:

Bytes 1-5
  UDDS control information (see “Using User-Defined Data Stream” on page 6-14)

Byte 6
  '6' (4700 control character)

Byte 7 (X'02')
  Function-management-header length

Byte 8
  Function-management-header identifier (X'80')

Bytes 9-10
  Function-management-header type (defined in CHX/VS Logic Manual)

Bytes 11-n
  Application data (defined in the CHX/3694 Logic Manual)

The data written by your application program must have this format if the data does not contain a function-management-header.

Bytes 1-5
  UDDS control information (see “Using User-Defined Data Stream” on page 6-14)

Byte 6
  '0' (4700 control character)

Bytes 7-n
  Application data (defined in the CHX/3694 Logic Manual)

To prepare to read the data from the 3694 processor without writing any data, your application program should do a write/read operation with the following data stream:

Bytes 1-5
  UDDS control information (see “Using User-Defined Data Stream” on page 6-14)

Bytes 1-2 (X'0002')
  Length of data being passed to the 3694 application is set to a minimum value of 2 bytes

Bytes 3-4 (X'0200')
  Length of data being received from the 3694 application is set to the maximum data length permitted

Byte 5 (X'73')
  UDDS control flag is set to a value representing a write/read operation

Byte 6
  '5' (4700 control character) invite device without writing

Note: Data is not actually sent to the 3694 processor. The data is only invited from that device.
Example of User-Defined Data Stream

Figure 6-17 shows a typical communications plan using UDDS.

<table>
<thead>
<tr>
<th>AS/400 Application Program</th>
<th>Finance Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>INIT-SELF</td>
</tr>
<tr>
<td>Request INIT-SELF</td>
<td>INIT-SELF Response</td>
</tr>
<tr>
<td>INIT-SELF</td>
<td>BIND</td>
</tr>
<tr>
<td>INIT-SELF Response</td>
<td>BIND Response</td>
</tr>
<tr>
<td>WRITE/READ UDDS</td>
<td>Start Data Traffic (SDT)</td>
</tr>
<tr>
<td></td>
<td>SDT Response</td>
</tr>
<tr>
<td></td>
<td>LU 0 Data Stream</td>
</tr>
<tr>
<td>WRITE/READ UDDS</td>
<td>TERM-SELF</td>
</tr>
<tr>
<td>TERM-SELF</td>
<td>TERM-SELF Response</td>
</tr>
<tr>
<td>TERM-SELF Response</td>
<td>UNBIND</td>
</tr>
<tr>
<td>CLOSE</td>
<td>UNBIND Response</td>
</tr>
</tbody>
</table>

Figure 6-17. UDDS Communications Scenario
Chapter 7. Finance Considerations

This chapter describes considerations for programming, controllers, performance, and Systems Network Architecture (SNA) for finance communications.

Intersystem Communications Function

This section discusses considerations when using ICF finance.

Programming Applications

When planning your programs for use with finance communications, you must consider the protocol being used, the error recovery objectives, and the performance objectives.

ICF finance uses a half-duplex contention protocol. Half-duplex means that information can be sent in one direction at a time over the data link. Only when the information transmission ends can information be sent in the opposite direction. Contention means that you cannot control which program should send and which program should receive. If your AS/400 application program and the remote controller program both attempt to send data at the same time, the AS/400 program is the contention loser. Then the AS/400 program receives the data from the remote controller program and sends the data when the controller program finishes sending data.

Consider the following when using programs for ICF finance communications:

- Ensure your program checks the major and minor codes after every operation to determine whether the operation was a success or a failure. For more information about the error codes, see Appendix B, “Return Codes, Messages, and Sense Codes.”
- Use the information in the input/output (I/O) feedback area for your program. For more information about the I/O feedback area, see “Using Input/Output Feedback Area” on page 5-10.
- Remember, if the AS/400 program is the target program, it cannot start error recovery. If a permanent error occurs, the target program should finish any needed processing and end. The controller program is responsible for establishing the session again.

Performance

If you experience performance problems, performance improvements could occur when additional storage is moved from the base pool to the machine pool. For additional information concerning ways to improve your system performance, see the book, Work Management.

Buffering: Finance communications support buffers data sent by your program. The first record of a group is always sent immediately. The remaining records in a group are not sent until the communications buffer becomes full, a force-data function is specified, or the last record in a group is sent. Only single record groups are sent to a 3694 processor. Data sent on a write operation always results in a single record group being sent to the 3694 processor, regardless of the functions specified on the write operation.

Responses: A group of records that was partially sent can be completed by explicitly specifying the end-of-group (ENDGRP) function. A group of records also can be implicitly completed by a read, write-with-read, or a write-with-invite function.

For a 47xx or FBSS controller, if a group of records ended explicitly with the end-of-group function, a response is required from the remote controller program. The write operation does not end until the response is received. This may not be desirable when performance is a critical consideration. A response is not required from a 3694 processor, regardless of the functions specified on the write operation.
When a group of records is closed implicitly by a read operation, a write-with-invite function, or a write-with-read operation, a response is not required for the group from the remote controller.

When a group of records is sent on the system monitor session for a 4701 or 4702 controller, a response is always required from the controller before the write operation ends.

**Prestart Jobs Using Program Start Requests:** To minimize the time required to start a job when a program start request is received, you can use a prestart job entry to start a job on the AS/400 system before the controller program sends a program start request.

To use prestart jobs and to ensure programming changes are made in the prestart job program, you must define both communications and prestart job entries in the subsystem description. For more information about the prestart job entries, see the book, *ICF Programming*.

**Program Initialization Parameters:** If a program start request is received, each parameter received must be equal in length to the corresponding parameter specified in the AS/400 program. If the received parameter length exceeds the parameter length in the AS/400 program, truncation occurs. If the received parameter length is less than the parameter length in the AS/400 program, results that cannot be predicted could occur.

**Security:** If the data supplied on the INIT-SELF command fails security checking for any reason, the session will not be established. The INIT-SELF command may also fail due to a previous session not being completely reset, or due to parameters that are not valid on the INIT-SELF request that is received. The INIT-SELF command will be rejected.

A message describing the error and the sense code is issued to the QSYSOPR message queue.

For more information about sense codes, see Appendix B, “Return Codes, Messages, and Sense Codes.”

---

### Non-Intersystem Communications Function

This section discusses considerations when using non-ICF finance.

### Programming Applications

The following programming considerations apply to the non-ICF finance communications functions using finance input/output manager (FIOM), Submit Finance Job (SBMFNCJOB) command, and user-defined data stream (UDDS).

**Finance Input/Output Manager (FIOM):**

When designing AS/400 application programs, you can use FIOM to do the following:

- To perform more than one write operation to the controller (rather than using the SBMFNCJOB command).
- To communicate interactively with the controller.
- To use a call interface (no communications operations).
- To write your own router program to handle data in a nonserial manner.

If you decide to use FIOM, you must override the QDFNDATA file by using the Override with Display File (OVRDSPF) command when defining devices to be used by the display file. Your application program must:

- Use the QDFNDATA file. If you do not, the FIOM routine receives a file-not-opened exception with CPD8289 (input/output exception received) and CPF8390 (errors occurred when running the program) messages. These messages are sent to your application program.
- Acquire and release finance devices used by the job, unless FIOM is being used with the SBMFNCJOB command.
- Open the QDFNDATA file as a shared file before a FIOM routine is started. The file should be closed by your application before returning control to the program that started your application. The SBMFNCJOB command opens and closes the file and acquires and releases the finance devices.
• Ensure the data length parameter passed to the QFNWRT, QFNWRTI, QFNREAD, and QFNREADI routines is initialized to a valid packed decimal number (15 5) to avoid receiving a CPD8286 (invalid format for data length parameter) error message.

• Ensure the data parameter is large enough to handle the maximum length of data that could be received from the finance controller application. A maximum of 512 bytes is allowed. See the XPF programmer’s guide and the controller application guide because there may be requirements of fewer than 512 bytes. If the data parameter is not long enough, adjacent data space could be overwritten with financial data and give unexpected results.

• Process the send and receive data according to the format defined by the controller application and to ensure that the application program conforms to SNA communications rules.

• Handle any error recovery because errors received by the FIOM routines result in error messages sent to your application program.

Submit Finance Job (SBMFNCJOB) Command: This command starts a continual BATCH job. The transaction programs receive controller data from and give data to the SBMFNCJOB command and send the data back to the controller. The SBMFNCJOB command handles incoming data from all devices serially. Only one transaction program can be running at one time.

Consider the following information about the SBMFNCJOB command when designing application programs:

• Avoid sending data directly from the application program to the controller application to help minimize finance job wait time. Let the finance job return data to the 4701 or 4702 application whenever possible.

• Start the QFNROUTE program directly, to provide interactive debugging of the application programs. The format of the command is:

```call pgm(osys/qfnroute) parm(device-table-name program-table-name user-table-name 'message-queue-name')```

If you use the SBMFNCJOB interface to communicate with a controller, you must consider the hardware configuration of your AS/400 system. You must decide what devices to use, the number of devices per controller, and how these displays communicate.

To help you with these decisions, consider the following:

• Decide which finance controller application program to use by determining the amount of work that can be unloaded from the AS/400 system to the controller. The following factors affect this decision:
  – The amount of function supplied by the various applications at the controller level
  – The quantity and speed (due to the hardware configuration of your system) of transactions in which you require data to be passed to the AS/400 system
  – The amount of storage required for the application
  – Performance information supplied with each application

• Minimize unnecessary device acquires during the finance job starting phase by carefully changing the device tables. By balancing and distributing the processing load in the most efficient manner for the operating environment, the transaction processing you do at a later time improves. When designing the device tables, consider dividing devices in one of the following ways:

  • By common functions, for example, placing all tellers on one job and all loan officers on another job.

  • By controller, for example, placing all the devices on one controller assigned to one finance job.

Remember a finance job must acquire the first device it specifies in a device table to successfully start the finance job. The finance device must be active and not be in use by another job.

If a device is included in more than one device table, and the SBMFNCJOB interface is used, only one finance job can acquire that device.
Consider the possibility for I/O failure in your environment. If an unrecoverable I/O error occurs on a finance controller or line, the job started by the SBMFNCJOB command ends. This also ends communication with all devices associated with that job. For more information about I/O error handling capabilities of the SBMFNCJOB interface, see “Input/Output” on page 6-3.

Consider submitting multiple finance jobs. The jobs submitted handle transaction requests serially. The SBMFNCJOB command calls your transaction processing application and waits for control to return before it can process another request from any device associated with the finance job. Therefore, submitting more than one finance job reduces jobs waiting in a queue because of serial processing within one job.

As more devices are added to a device table, the program access group for the finance job using that device table increases, primarily due to the increased number of I/O buffers associated with the job. Therefore, submitting more than one finance job also reduces the group size of a single job accessing the program.

- Change the QFNC subsystem, job queue, and class to suit the needs of your particular operating environment. For example, you can change the QFNC class running priority to balance the workload of your system.
- Use the JOBD parameter of the SBMFNCJOB command to specify a job description having routing data other than QFNC specified on the QFNC job description. This allows you to specify different classes, and thus different running priorities and time slices, for individual finance jobs. The QFNC class is EXCPTY (20) and TIME SLICE (2000).
- Change this wait time by using the Create Class (CRTCLS) control language (CL) command to create a class with the DFTWAIT parameter set to the wait time you want. Display file QDFNDATA has been created with the WAITFILE parameter of (*CLS); therefore, the maximum amount of time spent trying to acquire a finance device is determined by the class associated with the finance job. The QFNC class specifies a default wait time of 30 seconds. If you reduce this wait time, the finance job may not have the ability to acquire a device in environments in which many devices are acquired or released at the same time.

With high use of the system, performance improvements can occur when additional storage is moved from the base pool to the machine pool. For additional information concerning system adjustment, see the book, *Work Management*.

Finance support attempts error recovery if a finance job receives an I/O exception response. When an I/O exception response signals a finance job, the major/minor return code is retrieved from the message to determine the possibility of error recovery. Recoverable errors alert the finance job to try a recovery procedure. If the procedure is successful, processing continues normally. If the process is unsuccessful, the next action depends on the nature of the error.

Device errors result in the release of the affected device but other devices associated with the job continue processing. However, if recovery is not successful, controller or line errors end the job.

If you use the SBMFNCJOB command as the communications method between the AS/400 system and the controllers, special security exists, as discussed in the following considerations.
**Granting Authority to Finance Objects:** To keep financial information secure, the objects shipped with the system have restricted accessibility. Therefore, the following tasks must be completed before using the SBMFNCJOB interface.

- Use the Grant Object Authority (GRTOBJAUT) command to grant authority to the following users.

  Individual or group authorization to programmers who update the tables using the WRKDEVTBL, WRKPGMTBL, and WRKUSRTBL commands:

  ```
  GRTOBJAUT OBJ(QSYS/WRKDEVFLB) OBJTYPE(+CMD)
  USER(user-name) AUT(+CHANGE)
  GRTOBJAUT OBJ(QSYS/WRKPGMTB) OBJTYPE(+CMD)
  USER(user-name) AUT(+CHANGE)
  GRTOBJAUT OBJ(QSYS/WRKUSRTBL) OBJTYPE(+CMD)
  USER(user-name) AUT(+CHANGE)
  ```

  Individual or group authorization to operators who submit finance jobs and must be authorized to the SBMFNCJOB command:

  ```
  GRTOBJAUT OBJ(QSYS/SBMFCJOB) OBJTYPE(+CMD)
  USER(user-name) AUT(+CHANGE)
  ```

- Authorize the user profile QFNC access to any devices, programs, libraries, and files used by the finance jobs submitted with the SBMFNCJOB command. Specifying AUT(*CHANGE) is enough authority for these programs. An alternative to granting the required authority is to specify the QFNC profile as the owner of the objects.

- Determine if the finance support user ID sent in the data stream with the INIT-SELF is to be approved. If the user ID is approved, determine which user ID is allowed in each finance job. Use one user table for each finance job, one table for all finance jobs, or a combination of uniquely shared user tables. Define your finance user tables using the WRKUSRTBL command.

  For more information about the WRKUSRTBL command, see “Work with User Table Command” on page 3-6. Ensure that these programs do the following:

  - Accept and return data according to specifications defined by the finance controller application program.

  Once you develop your programs, describe which programs are to be used to process transactions by using the WRKPGMTBL command. Use one table for each finance job, one for all finance jobs, or a combination of unique and shared program tables. For more information about the WRKPGMTBL command, see “Work with Program Table Command” on page 3-7.

- Follow the security instructions described in “Security” on page 7-6.

**Additional Security Considerations:** To improve the security of your finance system, use the following guidelines:

- Submit jobs through the SBMFNCJOB command using the QFNC user profile. Similarly, QFNC owns the commands used for working with device, user, and program tables. The password of the QFNC user profile should remain secure.

- To avoid the possibility of external use, do not create a job description that does not refer to QFNC support.

- Use the Display Job Description (DSPJOBD) command to display the default job description for the SBMFNCJOB command. You can change attributes of this job description (job logging level) with the Change Job Description (CHGJOBD) command. Creating different job descriptions also restricts access to individual finance jobs.

- Use the work with table commands to restrict access to transaction processing programs and devices. For example, in Figure 7-1 on page 7-6, only Jones has access to Program Table, PROG X.

- Consider which library list that the finance job should use. The current library list of the user’s job that performed the SBMFNCJOB command becomes the library list used for that finance job.
User-Defined Data Stream (UDDS): A user-defined data stream (UDDS) is a data stream in which the user has defined and embedded all device control characters. Consider the following information about UDDS when designing application programs:

- Ensure that your program correctly formats the data stream. If it does not, results that cannot be predicted could occur.
- Ensure that you compile your program with an externally described file that does not contain the USRDFN keyword. Then run the program with an externally described display containing the USRDFN keyword.

The MSGQ parameter (on the SBMFNCJOB command) expected by QFNROUTE is a 20-character variable (a 10-character message queue name immediately followed by a 10-character library name). The parameter represents the qualified name of the message queue to which finance messages are to be sent.

Security

After you have created your device, user, and program tables, have the person responsible for security on your system save backup versions of the QFNDEVtbl, QFNUSRtbl, and QFNPGMTbl files in another library.

Have the person responsible for security add the names of the libraries containing your AS/400 transaction application programs, and the names of any additional libraries required by those programs, to the original library list of QFNC. (QFNC is the job description found in the QGPL library under which the finance jobs run.)

Have the person responsible for security grant authorities required by user profile QFNC for programs used by your finance job. QFNC must be authorized for the device descriptions so that the devices can be acquired by the finance jobs, along with your AS/400 application programs and the libraries in which they reside.

Controller Applications

The following must be considered when you write your own application programs for use on the 4701, 4702, or FBSS controller.

To communicate successfully with finance communications, your controller application program must do the following:

- Handle and respond to the following commands received from the AS/400 system:
  - Activate Physical Unit (ACTPU)
  - Activate Logical Unit (ACTLU)
  - Bind (BIND)
  - Request for Start Data Traffic (SDT)
  - Clear (CLEAR)
  - Unbind (UNBIND)
- Start a session by sending the INIT-SELF request to the AS/400 system.
- End a session by sending the TERM-SELF request to the AS/400 system.
- Do not use brackets or the Change-of-Direction (CD) command during a session.
- Do not require a Bid (BID) sequence at the beginning of the session.

The AS/400 system requires that any device configured as a 3278 or a 3279 must accept extended data streams. Be sure that the emulation program running on the 4701, 4702, or FBSS controller has extended data stream support. If it does not have extended data stream support, configure for a 3277 device.

The device type you specify in the device description determines the form of data stream sent to the finance controller. The data stream can be one of the following:

- LU0 for device type 3624, 3694, 4704, or *FNCICF
- LU1 for device type 3287
- LU2 for device type 3277, 3278, or 3279
LU0 data streams allow printer data to be included with display data. LU1 data streams are sent to printers. The LU2 data streams require you to create 3270-type application displays. Be aware that translation and emulation could make the LU2 approach run slower than the LU0 support.

A 4701 or 4702 controller can do 5250 emulation (with a 4701- or 4702-based, 5250 emulation package), or can allow attachment of a personal computer emulating a finance device. Contact your IBM remarketer for information about additional finance configuration options.

---

**Systems Network Architecture (SNA)**

This section discusses the **Systems Network Architecture (SNA)** considerations for ICF and non-ICF finance communications. SNA describes the layered logical structure, formats, protocols, and operational sequences used for transmitting information units through networks, as well as controlling the configuration and operation of networks. The following data formats are used by SNA for the INIT-SELF command.

**INIT-SELF Command Field Format**

The INIT-SELF request starts an SNA session. Figure 7-2 shows the format needed for the INIT-SELF command.

<table>
<thead>
<tr>
<th>Offset in Decimal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 through 2</td>
<td>Network service header: must be hex 010681.</td>
</tr>
<tr>
<td>3 through 12</td>
<td>Initiate header: must be hex 0040404040404040F3.</td>
</tr>
<tr>
<td>13</td>
<td>Length of destination logical unit name: must be hex 08</td>
</tr>
</tbody>
</table>

**3694 Document Processor with ICF:**

The 3694 processor follows the same procedure for initiation as for the other finance controllers, except the 3694 processor also sends the program name to be started on the INIT-SELF along with the security data. Then the program can communicate with the controller using the finance support.

On a secure AS/400 system, the INIT-SELF request that is received must include a valid user ID and password in the user data field of the INIT-SELF command. A program name is required in the destination logical unit (LU) field of the INIT-SELF.

If any data is included in the user data field of the INIT-SELF request, the field must be 10 bytes in length and formatted as shown in Figure 7-3 on page 7-8.
47xx and FBSS Finance Controllers with Intersystem Communications

Function: The INIT-SELF user data field is also used by 47xx and FBSS controllers. On a secure system, the received INIT-SELF request must contain a user ID and password and can also contain a library name. Figure 7-4 shows the format of the user data field.

Finance Controllers with Non-ICF Finance: For information about non-ICF, see “Sending Data from the Finance Controller to the AS/400 System” on page 6-6.

If the SBMFNCJOB interface is used, the INIT-SELF must use the format in Figure 7-2 on page 7-7 with the following exceptions:

- Restrictions do not exist for the name of the destination logical unit.
- The library name is not present in the user data field.

If the SBMFNCJOB interface is not used, the format is determined by the user program.

BIND Command Field Format

The BIND command is used to start a session and to establish what protocol is followed for the current session. Figure 7-5 shows the BIND parameters for ICF finance to be used for each controller.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>47xx and FBSS Controllers</th>
<th>4701 and 4702 System Monitor</th>
<th>3694 Document Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function management profile</td>
<td>04</td>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>Transmission profile</td>
<td>04</td>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>Primary logical unit protocol</td>
<td>B0</td>
<td>A0</td>
<td>10</td>
</tr>
<tr>
<td>Secondary logical unit protocol</td>
<td>B0</td>
<td>B0</td>
<td>30</td>
</tr>
<tr>
<td>Common protocol</td>
<td>4040</td>
<td>4040</td>
<td>4040</td>
</tr>
</tbody>
</table>

Figure 7-6 on page 7-9 shows the BIND parameters for non-ICF finance to be used for each controller.
<table>
<thead>
<tr>
<th>Protocol</th>
<th>4701 and 4702 Controllers</th>
<th>3694 Document Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function management profile</td>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>Transmission profile</td>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>Primary logical unit protocol</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Secondary logical unit protocol</td>
<td>B0</td>
<td>30</td>
</tr>
<tr>
<td>Common protocol</td>
<td>4000</td>
<td>4000</td>
</tr>
</tbody>
</table>
Appendix A. Language Operations, DDS Keywords, and System-Supplied Formats

This appendix provides information about the following:

- Valid communications operations supported by ICF finance
- Valid finance communications operations supported and the associated high-level language operations
- Data description specifications (DDS) processing keywords
- System-supplied formats

Using Language Operations

You can use ICF operations and high-level program languages to use finance communications. This discussion defines the operations used for finance and the differences in the language statements for C Set ++ for OS/400, ILE C/400, COBOL/400, and RPG/400 programming languages.

Intersystem Communications Function Operations

Figure A-1 provides a brief description of the ICF operations supported by finance communications.

<table>
<thead>
<tr>
<th>ICF Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Opens the ICF file.</td>
</tr>
<tr>
<td>Acquire</td>
<td>Establishes an ICF session between the application and the</td>
</tr>
<tr>
<td></td>
<td>remote location.</td>
</tr>
<tr>
<td>Get-attributes</td>
<td>Determines the status of the session.</td>
</tr>
<tr>
<td>Read</td>
<td>Obtains data from a specific session.</td>
</tr>
<tr>
<td>Read-from-invited-</td>
<td>Obtains data from any session responding to an invite function.</td>
</tr>
<tr>
<td>program-devices</td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td>Passes data records from the local program to the remote</td>
</tr>
<tr>
<td></td>
<td>program.</td>
</tr>
<tr>
<td>Write/Read</td>
<td>Allows a write operation followed by a read operation.</td>
</tr>
<tr>
<td>Release</td>
<td>Attempts to end an ICF session.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the ICF file.</td>
</tr>
</tbody>
</table>
**Intersystem Communications**

**Function Language Statements**

Figure A-2 provides a list of ICF operations supported by finance communications and the equivalent language statements needed to run these operations.

---

**Figure A-2. ICF Operations and Equivalent Language Statements**

<table>
<thead>
<tr>
<th>ICF Operation</th>
<th>ILE RPG/400 Function</th>
<th>ILE COBOL/400 Procedure Statement</th>
<th>C Functions$^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>OPEN</td>
<td>OPEN</td>
<td>fopen or _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 or _Rreadn</td>
</tr>
<tr>
<td>Read-from-invited-program-devices</td>
<td>READ$^1$</td>
<td>READ$^1$</td>
<td>fread or _Rreadindv</td>
</tr>
<tr>
<td>Write</td>
<td>WRITE</td>
<td>WRITE</td>
<td>fwrite or _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>_Rrelease</td>
</tr>
<tr>
<td>Close</td>
<td>CLOSE</td>
<td>CLOSE</td>
<td>fclose or _Rclose</td>
</tr>
</tbody>
</table>

**Notes:**

1. A read operation can be directed either to a specific program device or to any invited program device. The support provided by the compiler that 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 operation is issued with a format or display specified, the read operation is interpreted as an ICF read operation. Refer to the appropriate language reference book for more information.

2. $C$ Functions represents functions for both ILE C/400 and C Set ++ for OS/400 languages. Also, both languages are case sensitive.
Data Description Specifications

Keywords

Read and write operations use a record containing DDS keywords. These keywords allow you to use more specific communications functions with the read and write operations. Figure A-3 shows all the keywords supported by ICF finance communications.

<table>
<thead>
<tr>
<th>DDS Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANCEL</td>
<td>Cancels a group of records that was partially sent.</td>
</tr>
<tr>
<td>CNLINVITE</td>
<td>Cancels any valid invite for which data has not yet been received.</td>
</tr>
<tr>
<td>ENDGRP</td>
<td>Indicates the end of a user-defined group of records.</td>
</tr>
<tr>
<td>EOS</td>
<td>Specifies an end-of-session function.</td>
</tr>
<tr>
<td>FAIL</td>
<td>Sends a fail indication to the remote system.</td>
</tr>
<tr>
<td>FMH</td>
<td>Informs the remote program that a function-management-header is being sent.</td>
</tr>
<tr>
<td>FRCDTA</td>
<td>Sends data immediately for the write operation and also sends data currently in the communications buffer, without waiting for the buffer to become full.</td>
</tr>
<tr>
<td>INVITE</td>
<td>Schedules an invite request.</td>
</tr>
<tr>
<td>NEGRSP</td>
<td>Informs the remote system that the data received is not valid.</td>
</tr>
<tr>
<td>RCVENDGRP</td>
<td>Indicates that the end of a user-defined group (chain) of records was received.</td>
</tr>
<tr>
<td>RCVFMH</td>
<td>Indicates to the program that a function-management-header was received.</td>
</tr>
<tr>
<td>RCVNEGRSP</td>
<td>Indicates that the remote program sent a negative response.</td>
</tr>
<tr>
<td>RECID</td>
<td>Allows the data content to identify the record format to use for receiving data.</td>
</tr>
<tr>
<td>TIMER</td>
<td>Allows the user to specify an interval of time to wait before a read-from-invited-program-devices operation receives a timer-expired return code.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDS Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARLEN</td>
<td>Specifies that the length of the user data is defined in the 5 bytes of the specified field.</td>
</tr>
</tbody>
</table>

1 Not valid for a 3694 processor.

System-Supplied Formats

Figure A-4 shows the functions and operations performed by the system-supplied formats that are valid for finance communications.

<table>
<thead>
<tr>
<th>System-Supplied Format</th>
<th>Equivalent DDS Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$CANL</td>
<td>CANCEL, INVITE</td>
<td>Send SNA cancel, then invite</td>
</tr>
<tr>
<td>$$CNLINV</td>
<td>CANCEL</td>
<td>Send SNA cancel</td>
</tr>
<tr>
<td>$$CNLINV</td>
<td>CNLINVITE</td>
<td>Cancel an invite</td>
</tr>
<tr>
<td>$$EOS</td>
<td>EOS</td>
<td>End of session</td>
</tr>
<tr>
<td>$$FAIL</td>
<td>FAIL</td>
<td>Fail</td>
</tr>
<tr>
<td>$$NRSP</td>
<td>NEGRSP, INVITE</td>
<td>Negative response, then invite</td>
</tr>
<tr>
<td>$$NRSPNI</td>
<td>NEGRSP</td>
<td>Negative response</td>
</tr>
<tr>
<td>$$SEND</td>
<td>INVITE</td>
<td>Write then invite, or invite</td>
</tr>
<tr>
<td>$$SEnde</td>
<td>ENDGRP</td>
<td>Write with end-of-group</td>
</tr>
<tr>
<td>$$SENDFM</td>
<td>FMH, Invite</td>
<td>Write with function-management-header, then invite</td>
</tr>
<tr>
<td>$$SENDNF</td>
<td>FMH</td>
<td>Write with function-management-header</td>
</tr>
<tr>
<td>$$SENDNI</td>
<td>No DDS keyword</td>
<td>Write</td>
</tr>
<tr>
<td>$$TIMER</td>
<td>TIMER</td>
<td>Set timer</td>
</tr>
</tbody>
</table>

1 Not valid for a 3694 document processor.
Appendix B. Return Codes, Messages, and Sense Codes

Return Codes

This section describes all the return codes that are valid for finance 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 local AS/400 application program that issues the operation and receives a return code from ICF communications. The **remote program** refers to the application program on the remote system with which your program is communicating through ICF.
2. Several references to input and output operations are made in the descriptions. These operations can include DDS keywords and system-supplied formats, which are listed in Appendix A.

Major Code 00

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 0000  | **Description**: For output operations issued by your program, 0000 indicates that the last output operation completed successfully and that your program can continue to send data.  
**Action**: Issue an input or output request. |
| 0001  | **Description**: Your program has successfully invited the finance session.  
**Action**: Issue a read-from-invited-program-devices operation. |
0003  **Description:** On a successful input operation, your program received a group of records.

**Action:** Issue an input operation to receive the next group of records, or issue an output operation.

0007  **Description:** On a successful input operation, your program received a group of records with a function-management-header (FMH).

**Action:** Issue an input or output operation.

## Major Code 02

**Major Code 02** – Input operation completed successfully, but your job is being ended (controlled).

**Description:** The input operation issued by your program completed successfully. Your program may have received some data or a message from the remote system. However, your job is being ended (controlled).

**Action:** Your program should complete its processing and end as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

<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. Also, your job is being ended (controlled).

**Action:** Your program can continue to receive data, or it can send data to the remote 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.

| 0203   | **Description:** On a successful input operation, your program received a group of records. Also, your job is being ended (controlled).

**Action:** Your program can issue an input operation to receive the next group of records, or 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.

| 0207   | **Description:** On a successful input operation, your program received a group of records with a function-management-header (FMH). Also, your job is being ended (controlled).

**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.
## 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 for a function-management-header or a timer indication, and continue with the next operation.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 0303  | **Description:** On a successful input operation, your program received an end-of-group indication without any data.  
**Action:** Issue an input or output operation. |
| 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) |
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 or a response indication. The data from your output operation was not sent. You can attempt to send the data later.

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 0412   | **Description:** An output exception occurred because your program attempted to send data or to cancel an invite function when it should be receiving data or a response indication that was sent by the remote program. The data from your output operation was not sent to the remote system. Your program can attempt to send the data later.  
**Action:** Issue an input operation to receive the data or response indication.  
**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.

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

**Description:** The input operation attempted by your program was not successful. The data received was too long for your program's input buffer or was not compatible with the record format specified on the input operation.

**Action:** Refer to the minor code description for the appropriate recovery action.

<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 remote system was longer than the length specified for your program's input buffer. The length of the data record received from the remote 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) |
| 3441  | **Description:** A valid record format name was specified with format selection type *RECID. However, although the data received matched one of the record formats in the ICF file, it did not match the format specified on the read operation.  
**Action:** Correct your program to issue a read operation that does not specify a record format name, or specify the correct record format name to process the data based on the format selection option for the file.  
**Messages:**  
CPF5058 (Notify) |
| 3451  | **Description:** Your program specified a file record size that was not large enough for the indicators to be included with the data sent by the remote 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.

Code 8081 Description/Action

8081 Description: The operation attempted by your program was not successful because a system error condition was detected.

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

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

Messages:

CPF4170 (Escape)
CPF4510 (Escape)
CPF5197 (Escape)
CPF5244 (Escape)
CPF5257 (Escape)
CPF5274 (Escape)
CPF5346 (Escape)
CPF5355 (Escape)
8082 **Description:** The operation attempted by your program was not successful because the device supporting communications between your program and the remote location is not usable. For example, this may have occurred because communications were stopped for the device by a Hold Communications Device (HLDCMNDEV) command. Your program should not issue any operations to the device.

**Action:** Communications with the remote program cannot resume until the device has been reset to a varied on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off and then on again. Your program can attempt to establish the session again, continue local processing, or end.

**Messages:**
- CPF4744 (Escape)
- CPF5269 (Escape)

80B3 **Description:** The open operation issued by your program was not successful because the ICF file is in use by another process.

**Action:** Wait for the file to become available, then issue another open operation. Otherwise, your program may continue processing, or it can end.

Consider increasing the WAITFILE parameter with the Change ICF File (CHGICFF) or Override ICF File (OVRICFF) command to allow more time for the file resources to become available.

**Messages:**
- CPF4128 (Escape)

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)
- CPF4564 (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 remote program, you can:
- Continue processing without the session.
- End.

Several of the minor codes indicate that an error condition must be corrected by changing a value in the communications configuration or in the file.
- To change a parameter value in the communications configuration, vary the configuration off, make the change to the configuration description, then vary the configuration on.
- To change a parameter value in the file, use the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

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

Several other minor codes indicate a line or remote system error and may require an operator to correct the error.

**Note:** If the session is started again, it starts from the beginning, not at the point where the session error occurred.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 8140  | **Description:** A cancel reply was received from your program or from the operator in response to a notify message, or was the result of a system default, causing the session to be ended. The session is no longer active.  
**Action:** If your program started the session, issue an acquire operation to start the session again. If your program was started by a program start request, it can continue local processing or end.  
**Messages:** CPF5104 (Escape) |
| 8191  | **Description:** A permanent line or controller error occurred on an input or output operation, and the system operator attempted recovery in response to the error message. You can learn what type of line error occurred by checking the system operator's message queue. The session has ended. Data may have been lost.  
**Action:** If your program started the session, issue an acquire operation to start the session again. If your program was started by a program start request from the remote program, it can continue local processing or end.  
**Messages:** CPF4146 (Escape) CPF5128 (Escape) CPF5342 (Escape) CPF5344 (Escape) |
| 8197  | **Description:** On an input or output operation, the remote system ended the transmission abnormally because it could not continue the session. The session has ended.  
**Action:** If your program started the session, issue an acquire operation to start the session again. If your program was started by a program start request from the remote program, it can continue local processing or end.  
**Messages:** CPF5167 (Escape) CPF5241 (Escape) |
| 81A3  | **Description:** The session ended abnormally because of an SNA request shutdown, request recovery, or UNBIND command from the remote controller.  
**Action:** Determine the reason for the error in the remote controller program. Correct the error, then start the session again.  
**Messages:** CPF5167 (Escape) |
| 81A4  | **Description:** An SNA protocol violation occurred on the input or output operation attempted by your program. A negative-response with sense data was sent to the controller. |
Action: Examine the sense data in the associated message to determine the protocol error. Correct the error, then try the operation again. For more information on sense data, see the Systems Network Architecture Reference Summary book.

Messages:

CPF5248 (Escape)

81AD Description: The input or output operation issued by your program was not successful because the SDLC frame size was not large enough to contain the RU size. Either this was a configuration error, or the frame size was changed to a smaller value by the Exchange ID (XID) command.

Action: End your program. The SDLC frame size is specified in the MAXFRAME parameter on the controller description, and the RU size is specified in the MAXLENRU parameter on the device description. Verify that these configuration parameters are correct and, if necessary, reduce the RU size or increase the frame size. If changes to the configuration must be made, first vary the device off, then on again. Try to run your program again.

Messages:

CPF5341 (Escape)

81BA Description: Your program received a data record whose length exceeds the maximum user record length.

Action: Verify that the remote program sent the correct data. If so, end your program. Increase the value of the maximum record length (MAXRCDLEN) parameter in the ICF file, and increase the size of the input buffer on the record format to be used for the input operation, then try running your program again.

Messages:

CPF5205 (Escape)

81E9 Description: An input operation was issued and the format selection option for the ICF file was *RECID, but the data received did not match any record formats in the file. There was no format in the file defined without a RECID keyword, so there was no default record format to use. The session has ended.

Action: Verify that the data sent by the remote program was correct. If the data was not correct, have the operator on the remote system change the remote program to send the correct data. If the data was correct, add a RECID keyword definition to the file that matches the data, or define a record format in the file without a RECID keyword so that a default record format can be used on input operations. If your program started the session, use another acquire operation to start the session again. If a program start request started your program, continue local processing or end.

Messages:

CPF5291 (Escape)
Major Code 82

**Major Code 82** – Open or acquire operation failed.

**Description:** Your attempt to establish a session was not successful. The error may be recoverable or permanent, and recovery from it is unlikely until the problem causing the error is detected and corrected.

**Action:** You can perform the following general actions for all 82xx return codes. Specific actions are given in each minor code description.

If your program was attempting to start the session, you can:

- Correct the problem and attempt to establish the session again. The next operation could be successful only if the error occurred because of some temporary condition such as the communications line being in use at the time. If the operation is still not successful, your program should end.
- Continue processing without the session.
- End.

If your session was initiated by a program start request from the remote 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.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| 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 SYS) command was being issued.  
- The maximum number of prestart jobs (MAXJOBS parameter) was reduced by the Change Prestart Job Entry (CHGPJE) command.  
- The value for the maximum number of program start requests allowed (specified in the MAXUSE parameter on the ADDPJE or CHGPJE command) was exceeded.  
- Too many unused prestart jobs exist.  
- The prestart job had an initialization error.  
**Action:** Complete all processing and end your program as soon as possible. Correct the system error before starting this job again.  
**Messages:**  
CPF4292 (Escape)  
CPF5313 (Escape) |
| 8233 | **Description:** A program device name that was not valid was detected. Either an ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command was not run, or the program device name in your program does not match the program device name specified in the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE 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, CHGICFDEVE, or OVRICFDEVE command, specify the correct value in the PGMDEV parameter.  
**Messages:**  
CPF4288 (Escape)  
CPF5068 (Escape) |
| 8281 | **Description:** On an unsuccessful open or acquire operation, a system error condition was detected. For example, the file may previously have been in error, or the file could not be opened due to a system error.  
**Action:** Your communications configurations may need to be varied off and then on again. Your program can do one of the following:  
- Continue local processing.  
- Close the ICF file, open the file again, and acquire the program device again. However, if this results in another 8281 return code, your program should close the file and end.  
- Close the file and end.  
**Messages:**  
CPF4143 (Escape)  
CPF4168 (Escape)  
CPF4182 (Escape) |
8282  **Description:** The open or acquire operation attempted by your program was not successful because the device supporting communications between your program and the remote location is not usable. For example, this may have occurred because communications were stopped for the device by a Hold Communications Device (HLDCMNDEV) command. Your program should not issue any operations to the device. The session was not started.

**Action:** Communications with the remote program cannot resume until the device has been reset to a varied on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off, then on again. Your program can attempt to acquire the program device again, continue local processing, or end.

**Messages:**
- CPF4298 (Escape)
- CPF5269 (Escape)

8291  **Description:** A permanent line or controller error occurred on an unsuccessful open or acquire operation, and the system operator took a recovery option in response to the error message. The session was not started.

**Action:** If your program was attempting to start the session, it can try the acquire operation again. If your program was started by a program start request from the remote program, your program can continue local processing or end.

**Messages:**
- CPF4193 (Escape)
- CPF4261 (Escape)
- CPF5260 (Escape)
- CPF5342 (Escape)
- CPF5344 (Escape)

8297  **Description:** An SNA TERM-SELF or UNBIND request was received, while your program was attempting to establish a communications session with the remote controller.
**Action:** Try the open or acquire operation again, continue local processing, or end your program.

**Messages:**

CPF4178 (Escape)
CPF5167 (Escape)
CPF5241 (Escape)

82A2 **Description:** Your program issued an SNA INIT-SELF request for a finance remote location or device description that did not contain valid authorization data. One of the following occurred:

- The user ID or password was not supplied.
- The specified user ID was not found on the system.
- The specified password was not valid for this user ID.
- The specified user ID was not authorized to use this device description.

**Action:** Verify that your program specifies the correct user ID or password on the INIT-SELF command, or create a user profile with the correct user ID and password. If the user is not authorized to the device, use the Grant Object Authority (GRTOBJ) command to authorize the user.

**Messages:**

CPF4177 (Escape)
CPF5251 (Escape)

82A4 **Description:** An SNA protocol violation occurred on the open or acquire operation attempted by your program. A negative-response with sense data was sent to the controller.

**Action:** Examine the sense data in the associated message to determine the protocol error. Correct the error, then try the operation again. For more information on sense data, see the *Systems Network Architecture Reference Summary* book.

**Messages:**

CPF4141 (Escape)
CPF5248 (Escape)

82A6 **Description:** One of the following occurred:

- A negative-response with sense data was received when the Systems Network Architecture (SNA) BIND or Start Data Traffic (SDT) command was sent to the user to start the session.
- The BIND or SDT command did not end within the time limit specified using the wait time (WAITFILE) parameter on the CRTICFF, CHGICFF, or OVRICFF command.

The session was not started.

**Action:** Close the file. Examine the associated messages for SNA sense data received when the BIND or SDT command failed, and verify that the local and remote configurations are compatible. Determine why the remote controller did not respond within the time limit. If the time limit is too short, increase the value specified in the WAITFILE parameter on the CHGICFF or OVRICFF command. Correct the error and run the program again.
82A7 Description: The open or acquire operation attempted by your program was not successful because the specified program device was already in use. The session was not started.

Action: Your program can wait for the program device to become available, then try the open or acquire operation again. Otherwise, it can continue local processing or end.

Messages:
- CPF4142 (Escape)
- CPF4254 (Escape)
- CPF4333 (Escape)
- CPF4527 (Escape)
- CPF52/zerodot2 (Escape)
- CPF524/zerodot (Escape)
- CPF5538 (Escape)
- CPF4106 (Escape)
- CPF5507 (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 *REQUESTER device was released because an end-of-session was requested.
- The job does not have a *REQUESTER device; that is, the job was not started by a program start request.
- A permanent error occurred on the session.

Action:
- If the *REQUESTER device is already acquired and your program expects to communicate with the *REQUESTER device, use the program device that acquired the *REQUESTER.
- If 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 remote 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 RMTLOCNAME parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

Messages:

CPF4103 (Escape)
CPF4363 (Escape)
CPF4364 (Escape)
CPF4747 (Escape)
CPF5378 (Escape)
CPF5379 (Escape)

82AB Description: The open or acquire operation attempted by your program was not successful because the device description for the remote location was not varied on. The session was not started.

Action: Your program can wait until the communications configuration is varied on and then issue the acquire operation again, it can try the acquire operation again using a different device description, continue local processing, or end.

Messages:

CPF4285 (Escape)
CPF5333 (Escape)

82AD Description: The open or acquire operation attempted by your program to establish an SNA session was not successful because the SDLC frame size was not large enough to contain the RU size. Either this was a configuration error, or the frame size was changed to a smaller value by the Exchange ID (XID) command.
**Action:** End your program. The SDLC frame size is specified in the MAXFRAME parameter on the controller description, and the RU size is specified in the MAXLENRU parameter on the device description. Verify that these configuration parameters are correct and, if necessary, reduce the RU size or increase the frame size. If changes to the configuration must be made, first vary the device off, then on again. Try to run your program again.

**Messages:**

- CPF4260 (Escape)
- CPF5341 (Escape)

**82B3 Description:** The open or acquire operation attempted by your program was not successful because your program is trying to use a device description that is already in use by another job. The session was not started.

**Action:** Wait for the device description to become available, then issue the acquire operation again. You can use the Work with Configuration Status (WRKCFGSTS) command to determine which job is using the device description. Consider increasing the WAITFILE parameter of the CHGICFF or OVRICFF command to allow more time for the device to become available. Otherwise, your program can continue local processing or end.

**Messages:**

- CPF4282 (Escape)
- CPF5332 (Escape)

**82EA Description:** The open or acquire operation attempted by your program was not successful. A format selection of *RECID was specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, but cannot be used with the ICF file because the RECID DDS keyword is not used on any of the record formats in the file. The session was not started.

**Action:** Close the ICF file. Change the record format selection (FMTSLT) parameter to select formats by some means other than *RECID, or use a file that has a RECID DDS keyword specified for at least one record format. Open the file again.

**Messages:**

- CPF4348 (Escape)
- CPF5521 (Escape)

**82EC Description:** The acquire operation attempted by your program was not successful because finance communications does not support FMTSLT(*RMTFMT).

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

**Messages:**

- CPF4347 (Escape)
- CPF5515 (Escape)
82EE  Description: Your program attempted an open or acquire operation to a device that is not supported. Your program tried to acquire a device that is not a valid ICF communications type, or it is trying to acquire the requesting program device in a program that was not started by a program start request. The session was not started.

Action: Your program can continue local processing or end. Verify that the name of the remote location is specified correctly in the RMTLOCNAME parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. If your program was attempting to acquire a non-ICF device, use the appropriate interface for that communications type. If your program was attempting to acquire a requesting program device, verify that your program is running in the correct environment.

Messages:

- CPF4105 (Escape)
- CPF4223 (Escape)
- CPF4251 (Escape)
- CPF4760 (Escape)
- CPF5038 (Escape)
- CPF5550 (Escape)

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

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

Messages:

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

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 remote system or because the remote system was not active at the time, a second attempt may be successful. If the operation is still not successful, your program should end the session.
- Issue an end-of-session function and continue processing without the session.
- End.

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

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

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

If no changes are needed in your file or in the configuration, and depending on what the return code description says, you should notify the remote location that a change is required at that location to correct the error received.

<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.</td>
</tr>
</tbody>
</table>
**Action:** Verify that your program does not attempt any operations without an active session. Also verify that your program correctly handles the permanent error or session-not-acquired return codes (80xx, 81xx, 82xx) it received on previously issued input and output operations. To recover from an incorrectly handled error condition, your program may or may not be able to issue another acquire operation, depending on the return code.

**Messages:**
- CPD4079 (Diagnostic)
- CPF4739 (Status)
- CPF5067 (Escape)
- CPF5068 (Escape)
- CPF5070 (Escape)

**8319 Description:** The remote program sent a negative-response with sense data.

**Action:** Examine the sense data in the I/O feedback area to determine the necessary error recovery.

**Messages:**
- CPF4813 (Notify)
- CPF4814 (Notify)

**831B Description:** Your program tried to specify invalid sense data on a negative-response function. Correct your program so 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, finance communications automatically sends 08110000 to the remote program.

**Messages:**
- CPF4820 (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 remote 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 remote 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 finance 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 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.
- A function-management-header function was issued with zero data length.
- A function-management-header function was issued, and it was not specified in the first record of a group.

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 remote 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 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:

- CPF4564 (Escape)
- CPF4764 (Notify)
- CPF4766 (Notify)
- CPF4790 (Notify)
- CPF4803 (Notify)
- CPF5132 (Escape)
- CPF5149 (Escape)

831F  Description:  Your program specified data or a length for the operation that was not valid; however, the session is still active. One of the following caused the error indication:

- On an output operation, your program tried to send a data record that was longer than the MAXRCDLEN value specified for the ICF file.
- The program used a read or write operation that specified a data length greater than the record format in the ICF file.
- If this was a timer function, the format of the timer interval was not HHMMSS.
- If a system-defined format was used to specify the operation, or if the variable-length-data-record (VARLEN) function was used, then the length of the user buffer was not valid.
Action: If you want your program to recover, try the operation again with a smaller data length. If you do not need your program to recover immediately, do one of the following:

- Change the record format length in the ICF file, or change the record length in your program and compile your program again.
- For an input operation, specify a data length equal to or less than the record format length, or do not specify a length at all.
- If the timer function was used, verify that the format of the timer interval is HHMMSS.
- For an output operation that used the variable-length-data-record (VARLEN) function, verify that the length specified is less than the record length specified for the ICF file when it was opened.

Messages:

- CPF4762 (Notify)
- CPF4765 (Notify)
- CPF4767 (Notify)

8322 Description: Your program tried to issue a negative-response or fail function. However, these operations are not valid at the current time.

- Your program can issue a negative-response function only when it has received data that was in error. In this case, it can issue the negative-response function on the next operation.
- Your program can issue a fail function only if it is attempting to send data or if it has received data for which an error indication can be sent.

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 not valid operation.

Messages:

- CPF4817 (Notify)

8323 Description: Your program attempted to issue a cancel function when data or a negative-response indication 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 not valid operation.

Messages:

- CPF4776 (Notify)

8326 Description: Your program attempted to issue 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 not valid operation.

**Messages:**

CPF4779 (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 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 release operation issued by your program was not successful because your program attempted the operation while the current transaction was still active, or the release operation issued by your program was not successful because of one of the following:

- The group of records sent by your program was not closed.
- Data is available for your program to receive.
- Sense data is available for your program to receive.

The release operation is not valid if a group of records was partially sent or received, or if any data or a negative-response indication were received from the remote program for which your program did not issue an input operation. The operation was not performed, but the session is still active.

**Action:** Use the detach function to end the current transaction before issuing a release operation. Correct the error that caused your program to issue an invoke function during an active transaction; then run your program again.

**Messages:**

CPF4819 (Notify)
CPF5099 (Notify)
83B6  **Description:** On an output operation, your program received an indication that the remote program has quiesced the SNA session on which this transaction is running by issuing the SNA quiesce-at-end-of-chain (QEC) command. The remote program may release the quiesced state at a later time by issuing the SNA release-quiesce command.

**Action:** Your program can wait and try the output operation again at a later time. Otherwise, your program can end the session, continue local processing, or end.

**Messages:**

CPF4816 (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)
Program Start Request Errors

When a program start request is rejected by the system, message CPF1269 is sent. This message contains information that can be used to determine why the program start request was rejected. Message CPF1269 is sent to the QSYSMSG message queue if that queue exists and is not damaged. If the QSYSMSG message queue is damaged or does not exist, the message is sent to the QSYSOPR message queue.

Figure B-1 shows the reason codes for rejected program start requests. This information is sent using CPF1269 to the system that issued the rejected program start request. Program start requests that request a response fail with the listed negative response sense codes; those that do not request a response fail with the listed LUSTAT sense codes.

<table>
<thead>
<tr>
<th>Reason Code</th>
<th>Negative Response Sense Code</th>
<th>LUSTAT Sense Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>084B0000</td>
<td>00001004</td>
<td>Attach request received for a device that is not allocated to an active subsystem.</td>
</tr>
<tr>
<td>402</td>
<td>084B0000</td>
<td>00001003</td>
<td>Requested device currently held by a Hold Communications Device (HLDCMNDEV) command.</td>
</tr>
<tr>
<td>403</td>
<td>084B0000</td>
<td>00001004</td>
<td>User profile cannot be accessed.</td>
</tr>
<tr>
<td>404</td>
<td>084B0000</td>
<td>00001004</td>
<td>Job description cannot be accessed.</td>
</tr>
<tr>
<td>405</td>
<td>084B0000</td>
<td>00001004</td>
<td>Output queue cannot be accessed.</td>
</tr>
<tr>
<td>406</td>
<td>084B0000</td>
<td>00001004</td>
<td>Maximum number of jobs defined by subsystem description are already active.</td>
</tr>
<tr>
<td>407</td>
<td>084B0000</td>
<td>00001004</td>
<td>Maximum number of jobs defined by communications entry are already active.</td>
</tr>
<tr>
<td>408</td>
<td>084B0000</td>
<td>00001004</td>
<td>Maximum number of jobs defined by routing entry are already active.</td>
</tr>
<tr>
<td>409</td>
<td>084B0000</td>
<td>00001004</td>
<td>Library on library list is in use exclusively by another job.</td>
</tr>
<tr>
<td>410</td>
<td>084B0000</td>
<td>00001004</td>
<td>Group profile cannot be accessed.</td>
</tr>
<tr>
<td>411</td>
<td>084B0000</td>
<td>00001004</td>
<td>Insufficient storage in machine pool to start job.</td>
</tr>
<tr>
<td>412</td>
<td>08120000</td>
<td>00001004</td>
<td>System values not accessible.</td>
</tr>
<tr>
<td>501</td>
<td>084C0000</td>
<td>00001000</td>
<td>Job description is not found.</td>
</tr>
<tr>
<td>502</td>
<td>084C0000</td>
<td>00001000</td>
<td>Output queue is not found.</td>
</tr>
<tr>
<td>503</td>
<td>084C0000</td>
<td>00001004</td>
<td>Class is not found.</td>
</tr>
<tr>
<td>504</td>
<td>084C0000</td>
<td>00001001</td>
<td>Library on library list is not found.</td>
</tr>
<tr>
<td>505</td>
<td>084C0000</td>
<td>00001000</td>
<td>Job description or job description library is damaged.</td>
</tr>
<tr>
<td>506</td>
<td>084C0000</td>
<td>00001001</td>
<td>Library on library list is destroyed.</td>
</tr>
<tr>
<td>507</td>
<td>084C0000</td>
<td>00001001</td>
<td>Duplicate libraries are found on library list.</td>
</tr>
<tr>
<td>508</td>
<td>084C0000</td>
<td>00001000</td>
<td>Defined size of storage pool is zero.</td>
</tr>
<tr>
<td>602</td>
<td>10010000</td>
<td>00001000</td>
<td>Value of transaction program name is reserved but not supported.</td>
</tr>
<tr>
<td>Reason Code</td>
<td>Negative Response Code</td>
<td>LUSTAT Sense Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>604</td>
<td>10010000</td>
<td>00001004</td>
<td>Matching routing entry is not found.</td>
</tr>
<tr>
<td>605</td>
<td>10010000</td>
<td>00001009</td>
<td>Program is not found.</td>
</tr>
<tr>
<td>704</td>
<td>080F0000</td>
<td>00001000</td>
<td>Password is not valid.</td>
</tr>
<tr>
<td>705</td>
<td>080F0000</td>
<td>00001000</td>
<td>User is not authorized to device.</td>
</tr>
<tr>
<td>706</td>
<td>080F0000</td>
<td>00001000</td>
<td>User is not authorized to subsystem description.</td>
</tr>
<tr>
<td>707</td>
<td>080F0000</td>
<td>00001000</td>
<td>User is not authorized to job description.</td>
</tr>
<tr>
<td>708</td>
<td>080F0000</td>
<td>00001000</td>
<td>User is not authorized to output queue.</td>
</tr>
<tr>
<td>709</td>
<td>080F0000</td>
<td>00001000</td>
<td>User is not authorized to program.</td>
</tr>
<tr>
<td>710</td>
<td>080F0000</td>
<td>00001000</td>
<td>User is not authorized to class.</td>
</tr>
<tr>
<td>711</td>
<td>080F0000</td>
<td>00001008</td>
<td>User is not authorized to library on library list.</td>
</tr>
<tr>
<td>712</td>
<td>080F0000</td>
<td>00001000</td>
<td>User is not authorized to group profile.</td>
</tr>
<tr>
<td>713</td>
<td>080F0000</td>
<td>00001000</td>
<td>User ID is not valid.</td>
</tr>
<tr>
<td>723</td>
<td>080F0000</td>
<td>00001000</td>
<td>There is no password associated with the user ID.</td>
</tr>
<tr>
<td>726</td>
<td>080F0000</td>
<td>00001000</td>
<td>User profile is disabled.</td>
</tr>
<tr>
<td>801</td>
<td>084C0000</td>
<td>00001000</td>
<td>More than 2000 bytes of program initialization parameters received for the prestart job.</td>
</tr>
<tr>
<td>802</td>
<td>084C0000</td>
<td>00001000</td>
<td>Subsystem ending in progress.</td>
</tr>
<tr>
<td>803</td>
<td>084B0000</td>
<td>00001004</td>
<td>Prestart job is either not active or is ending.</td>
</tr>
<tr>
<td>804</td>
<td>084B0000</td>
<td>00001004</td>
<td>WAIT(&quot;NO) specified on prestart job entry.</td>
</tr>
<tr>
<td>805</td>
<td>084B0000</td>
<td>00001004</td>
<td>MAXJOBS on prestart job entry exceeded.</td>
</tr>
<tr>
<td>806</td>
<td>084B0000</td>
<td>00001004</td>
<td>Prestart job ended too soon.</td>
</tr>
<tr>
<td>901</td>
<td>084B0000</td>
<td>00001000</td>
<td>Program initialization parameters are not valid.</td>
</tr>
<tr>
<td>902</td>
<td>084B0000</td>
<td>00001000</td>
<td>Number of parameters for program is not valid.</td>
</tr>
<tr>
<td>903</td>
<td>084B0000</td>
<td>00001000</td>
<td>Program initialization parameters required but not sent.</td>
</tr>
<tr>
<td>1001</td>
<td>08640000</td>
<td>00001000</td>
<td>System logic error; function check or unexpected return code encountered.</td>
</tr>
<tr>
<td>1002</td>
<td>08640000</td>
<td>00001000</td>
<td>System logic error; function check or unexpected return code encountered while receiving initialization parameters.</td>
</tr>
<tr>
<td>1501</td>
<td>084C0000</td>
<td>00001000</td>
<td>System/36 environment library not found.</td>
</tr>
<tr>
<td>1502</td>
<td>084C0000</td>
<td>00001000</td>
<td>Library QSSP not found.</td>
</tr>
<tr>
<td>1503</td>
<td>084C0000</td>
<td>00001000</td>
<td>File QS36PRC not found in library QSSP.</td>
</tr>
<tr>
<td>1504</td>
<td>084C0000</td>
<td>00001000</td>
<td>Procedure name is greater than 8 characters.</td>
</tr>
<tr>
<td>1505</td>
<td>084C0000</td>
<td>00001000</td>
<td>Not authorized to current library.</td>
</tr>
<tr>
<td>1506</td>
<td>084C0000</td>
<td>00001000</td>
<td>Not authorized to file QS36PRC in current library.</td>
</tr>
<tr>
<td>Reason Code</td>
<td>Negative Response Sense Code</td>
<td>LUSTAT Sense Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>1510</td>
<td>080F0000</td>
<td>00001000</td>
<td>Not authorized to procedure in current library.</td>
</tr>
<tr>
<td>1511</td>
<td>080F0000</td>
<td>00001008</td>
<td>Not authorized to System/36 environment library.</td>
</tr>
<tr>
<td>1512</td>
<td>080F0000</td>
<td>00001000</td>
<td>Not authorized to file QS36PRC in System/36 environment library.</td>
</tr>
<tr>
<td>1513</td>
<td>080F0000</td>
<td>00001000</td>
<td>Not authorized to procedure in System/36 environment library.</td>
</tr>
<tr>
<td>1514</td>
<td>080F0000</td>
<td>00001008</td>
<td>Not authorized to library QSSP.</td>
</tr>
<tr>
<td>1515</td>
<td>080F0000</td>
<td>00001000</td>
<td>Not authorized to file QS36PRC in library QSSP.</td>
</tr>
<tr>
<td>1516</td>
<td>080F0000</td>
<td>00001000</td>
<td>Not authorized to procedure in file QS36PRC in library QSSP.</td>
</tr>
<tr>
<td>1517</td>
<td>08640000</td>
<td>00001000</td>
<td>Unexpected return code from System/36 environment support.</td>
</tr>
<tr>
<td>1518</td>
<td>10010000</td>
<td>00001009</td>
<td>Problem phase program not found in library QSSP.</td>
</tr>
<tr>
<td>1519</td>
<td>080F0000</td>
<td>00001000</td>
<td>Not authorized to problem phase program in library QSSP.</td>
</tr>
<tr>
<td>1520</td>
<td>084B0000</td>
<td>00001004</td>
<td>Maximum number of target programs started (100 per System/36 environment).</td>
</tr>
<tr>
<td>2651</td>
<td>10010000</td>
<td>00001009</td>
<td>*EXEC statement not specified.</td>
</tr>
<tr>
<td>2652</td>
<td>10010000</td>
<td>00001009</td>
<td>Blank missing after *EXEC statement.</td>
</tr>
<tr>
<td>2653</td>
<td>10060002</td>
<td>00001009</td>
<td>Program name missing.</td>
</tr>
<tr>
<td>2654</td>
<td>10020000</td>
<td>00001009</td>
<td>Program name greater than 10 characters.</td>
</tr>
<tr>
<td>2655</td>
<td>10020000</td>
<td>00001001</td>
<td>Library name greater than 10 characters.</td>
</tr>
</tbody>
</table>
Appendix C. Mapping Intersystem Communications Function Operations to Systems Network Architecture Commands

This appendix shows the association between some of the Systems Network Architecture (SNA) commands and ICF finance communications application operations.

In Figure C-1, the ICF operation appears with a corresponding interaction between the AS/400 system and the finance controller with a finance device varied on. The device is varied on before this scenario begins.

<table>
<thead>
<tr>
<th>Application Program</th>
<th>Finance Communications</th>
<th>Finance Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td></td>
<td>INIT-SELF</td>
</tr>
<tr>
<td>ACQUIRE</td>
<td></td>
<td>INIT-SELF</td>
</tr>
<tr>
<td>WRITE Data 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE Data 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE/ENDGRP Data 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE/INVITE Data 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read-from-invited/devices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELEASE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN</td>
<td>+ RSP</td>
<td></td>
</tr>
<tr>
<td>ACQUIRE</td>
<td>+ RSP</td>
<td></td>
</tr>
<tr>
<td>WRITE Data 1</td>
<td>+ RSP</td>
<td></td>
</tr>
<tr>
<td>WRITE Data 2</td>
<td>+ RSP</td>
<td></td>
</tr>
<tr>
<td>WRITE/ENDGRP Data 3</td>
<td>+ RSP</td>
<td></td>
</tr>
<tr>
<td>WRITE/INVITE Data 4</td>
<td>+ RSP</td>
<td></td>
</tr>
<tr>
<td>Data 1 Data 2 Data 3 RC-0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 1 Data 2 Data 3 RC-0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 1 Data 2 Data 3 RC-0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 1 Data 2 Data 3 RC-0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 1 Data 2 Data 3 RC-0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Middle Chain Data 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>End Chain, RQD Data 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Only Chain, RQE Data 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Only Chain, RQE Data 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Only Chain, RQE Data 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Only Chain, RQE Data 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Only Chain, RQE Data 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ RSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELEASE</td>
<td>+ RSP</td>
<td>TERM-SELF</td>
</tr>
<tr>
<td>CLOSE</td>
<td>+ RSP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure C-1. SNA Commands with Corresponding ICF Application Operations
In Figure C-2, the SNA session is initiated by a controller application program.

Figure C-2. SNA Commands with Corresponding ICF Application Operations on an AS/400 System-Initiated SNA Session
Appendix D. 4701 Finance Controller Diskette Download

A 4701 finance controller must have an operating or initial program-load (IPL) diskette before it can be used. This diskette contains the basic license internal code, the controller configuration, and the application programs that can be created on a System/370 computer configured to support the 4700 finance communications system. The operating image created by the System/370 computer is loaded on the AS/400 system, and the file is sent to the 4701 controller using the system monitor session. This appendix describes the support that sends the file to the controller after it is loaded on the system.

When the 4701 controller receives the file, the controller uses the file to create an operational diskette that it later uses during its own IPL procedure. The procedure for sending an operational diskette image is usually done only for the following actions:

- Installing the 4701 controller
- Changing the 4701 application programs
- Upgrading the 4701 controller

Using the Send Finance Diskette Image Command

The download support can be accessed through the Send Finance Diskette Image (SNDFNCIMG) command. This command includes the following information:

The SNDFNCIMG command uses the file, member, and remote location name.

**File**

*Specifies the library and the file where the diskette image resides.*

**Remote location name**

*Specifies the remote location name of a finance device specified as TYPE(*FNCICF). This device must be attached to a 4701 finance controller with an 8-inch diskette drive or a 3601 controller configured as a 4701 controller. The local location address (the address of the logical unit) of the device must be 01.*

**Member**

*Specifies the member in the file containing the diskette image which was blocked into a basic exchange format.*

Have the person responsible for security authorize your user profile to the SNDFNCIMG command and the QCRFDWNLD ICF file. Use the Grant Object Authority (GRTOBJAUT) to grant authority.

GRTOBJAUT OBJ(QSYS/SNDFNCIMG) OBJTYPE(+CMD)
USER(user-name) AUT(+CHANGE)

GRTOBJAUT OBJ(QSYS/QCRFDWNLD) OBJTYPE(+FILE)
USER(user-name) AUT(+CHANGE)

To send the image, do the following:

1. Do an IPL of the controller with the operating diskette you are currently using.
2. Prepare the controller to accept the image. Start the system monitor on the controller and issue the 999 command to create a diskette.
3. Vary on the system monitor to be used.
   **Note:** All devices attached to the controller description must be varied off before the system monitor device is varied on.
4. Enter the SNDFNCIMG command.
The SNDFNCIMG command creates the original image again and sends the operating image to the controller. The controller builds the operating diskette by writing the operating image on a blank diskette. When this is done, you can use the diskette for the IPL procedure for the controller.

Figure D-1 shows the basic format of the diskette image and the basic exchange files.

If you cannot get the basic exchange file, you can create the file with the following procedure:

1. Create a diskette image file using the Host Diskette Image Create (HDIC) program. This program is a part of 4700 Finance Communications System Host Support for an IBM System/370 computer, 3031, 3032, 3033, or 4300. The diskette image file must be converted into a basic exchange file that can be sent by the SNDFNCIMG command.

2. Convert each 256-byte record from the diskette image into four 64-byte records, shown in Figure D-1.

3. Write the records to the basic exchange file. The required format for each data record is shown in Figure D-2 on page D-3.

Each record can be from 80 to 96 bytes in length. You can use positions 1 through 8 for an optional header or comments. You can use positions 80 through 96, if needed, for optional comments.

The SNDFNCIMG command uses the sequential block number and sequential record number to ensure the correct sequence when the command processes the file.

The first record in the file is an optional comment record. You can write any information in this record to identify the file. The remaining records contain the data from the diskette image file.
The data field contains 64 bytes of data from the diskette image file.

For more information about the diskette downloading support, see the IBM 4700 Finance Communications System: Subsystem Operating Procedures, and the Host Support User's Guide.

Figure D-2. Format Required for Data Records
Appendix E. Intersystem Communications Function Finance Example Programs

This appendix provides examples of COBOL/400, RPG/400, and ILE C/400 programs to demonstrate how finance communications is used.

Figure E-1 on page E-2 shows representations of example programs.

COBOL/400 Source Program for Local System

This section describes the objects needed on the local system to run the ILE COBOL/400 account inquiry finance program.

Configuration

The following configuration commands are used to create the synchronous data link control (SDLC) line, controller, and device descriptions used by the local system.

```plaintext
CRTLNSDLC LIND(FNCLINE) RSRCNAME(LIN/zerodot22) ONLINE(/c5197NO) ROLE(/c5197PRI) NRZI(/c5197YES) CRTCTLFNC CTLD(FNCCTL) TYPE(47/zerodot2) MODEL(/zerodot) LINKTYPE(/c5197SDLC) ONLINE(/c5197NO) LINE(FNCLINE) STNADR(/zerodot1) CRTDEVFNC DEVD(K/zerodot/zerodot1DEV) TYPE(/c5197FNCICF) LOCADR(/zerodot3) RMTLOCNAME(K/zerodot/zerodot1DEV) ONLINE(/c5197NO) CTL(FNCCTL)
```

Program Files

The following files are used by the local system.

K001ICF

The ICF file used to send and receive records from the finance controller. This file was created by using the following command:

```
crticff file(FNCLIB/K001ICF) srchfile(FNCLIB/QDDSSRC) srcmbr(K001ICF)
```

The following command defines the program device entry:

```
addicfdeve file(FNCLIB/K001ICF) pgmdev(FNCTRGT) rmtlocname(+REQUESTER)
```

An OVRICFDEVE command with the same parameters can also be used.

K001DBF

The database file that holds the account records. This file was created by using the following command:

```
crtpf file(FNCLIB/K001DBF) srchfile(FNCLIB/QDDSSRC) srcmbr(K001DBF)
```

K001PRT

The printer file used to format output to a printer. This file was created by using the following command:

```
crtprt file(FNCLIB/K001PRT) srchfile(FNCLIB/QDDSSRC) srcmbr(K001PRT)
```
Figure E-1. Example for ICF Finance Network

Figure E-2 (Part 1 of 2). DDS Source for ICF File K001ICF

[Source code and diagram details]

Figure E-2 (Part 2 of 2). DDS Source for ICF File K001ICF

[Source code and diagram details]
Figure E-3 (Part 1 of 2). DDS Source for Database File K001DBF

Expanded Source

Figure E-3 (Part 2 of 2). DDS Source for Database File K001DBF
Figure E-4. DDS Source for Printer File K001PRT
**Program Explanation:** The following explains the COBOL/400 account inquiry program example in Figure E-5 on page E-6.

1. This section of the program defines the database file (K001DBF), the ICF file (K001ICF), and the printer file (K001PRT) used in the program.

   K001DBF is the database file that contains the customer account information.

   K001ICF is the ICF file that sends records to and receives records from the online terminal support (OTS) application program on the finance controller.

   K001PRT is the printer file that sends communication error information to a printer device from the account inquiry program on the AS/400 system.

2. The ERROR-SECTION section of the program defines the error handling procedure for I/O errors on the K001ICF file. It is automatically called when an exception occurs while the program is running. Feedback data is moved to a printer record (ERRREC) and the record is printed. The session is ended by releasing the finance device. The files are then closed and the program is ended.

3. The program opens the files to be used. The program device (FNCTRGT) used by the program is explicitly acquired. This program device was previously added to the ICF file (K001ICF) by the ADDICFDEVE command.

4. The parameters passed to the account inquiry program by the finance controller are placed in a data structure. If the account number is not among the parameters passed, a message is sent to the finance controller and the program is ended.

5. A read operation is done from the database file (with the account number as the key).

6. If the account number is not in the database file, a message is sent to the finance controller and the program is ended. Otherwise, the account inquiry is a valid inquiry, and control is passed to 7 to process the transaction.

7. This routine sends the inquiry reply in two records. The first record contains customer information, and the second record contains account information.

   **Note:** The finance controller supports an RU size of 256 bytes. Therefore, if the inquiry reply was to be sent using one record, the data separators (hex ‘FFFFFFFF’) in the customer information data structure (FORMATTED-DATA-2) must be removed, and the 16-byte finance header must be inserted again after the first 256 bytes.

8. This routine is called to build and send the error messages to the finance controller by passing control to 11.

9. This routine is called to build and send the customer information to the finance controller by passing control to 11.

10. This routine is called to build and send the account information to the finance controller by passing control to 11.

11. A write operation with the account inquiry response is sent to the program device that sent the inquiry.

   **Note:** For performance reasons, the INVITE keyword is specified as part of VARREC format. For more information, see “Responses” on page 7-1.

12. This routine is called to end the program. The session is ended by releasing the finance device and closing the files.
Program . . . . . . . . . . . . . : K/zerodot/zerodot1
Library . . . . . . . . . . . . . : FNCLIB
Source file . . . . . . . . . . . . : PGMSRC
Library . . . . . . . . . . . . . : FNCLIB
Source member . . . . . . . . . . : K/zerodot/zerodot1 /zerodot7/17/89 14:12:37
Generation severity level . . . . : 29
Text 'description' . . . . . . . . . : "BLANK"
Source listing options . . . . . . : "NONE"
Generation options . . . . . . . : "NONE"
Message limit:
Number of messages . . . . . . . . : "NOMAX"
Message limit severity . . . . . : 29
Print file . . . . . . . . . . . . . : QSYSPRT
Library . . . . . . . . . . . . . : /c5197/LIBL
FIPS flagging . . . . . . . . . . : /c5197/NOFIPS /c5197/NOSEG /c5197/NODEB /c5197/NOOBSOLETE
SAA flagging . . . . . . . . . . : /c5197/NOFLAG
Flagging severity . . . . . . . . : /zerodot
Replace program . . . . . . . . . : /c5197/YES
Target release . . . . . . . . . . : /c5197/CURRENT
User profile . . . . . . . . . . . : /c5197/USER
Authority . . . . . . . . . . . . . : /c5197/LIBCRTAUT
Compiler . . . . . . . . . . . . . : IBM AS/4/zerodot/zerodot COBOL/4/zerodot/zerodot

Figure E-5 (Part 1 of 11). COBOL/400 Program
Figure E-5 (Part 2 of 11). COBOL/400 Program
Figure E-5 (Part 3 of 11). COBOL/400 Program
STMT SEQNR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7
IDENTFCN S COPYNAME CHG DATE
117 FNC-OUTPUT-DATA.
118 FNC-CONTROL.

/zerodot124/zerodot/zerodot /zerodot1 FNC-OUTPUT-DATA.
/zerodot125/zerodot/zerodot /zerodot5 FNC-CONTROL.
/c5197 THE TERMINAL-MODE VARIABLE PUTS THE 47 DISPLAY INTO LARGE SCREEN MODE WHEREBY 192 CHARACTERS ARE DISPLAYED, THEREBY ALLOWING MORE INFORMATION TO BE DISPLAYED, WHEN THE VARIABLE IS SET TO 1.

119 1/zerodot TERMINAL-MODE PIC 9 VALUE IS 1/zerodot.
12/zerodot FILLER PIC X(4) VALUE SPACES.
123 1/zerodot5 FNC-FORMATTED-DATA PIC X(24).

124 FORMATTED-DATA-1.
126 TEXT-1 PIC X(30).
127 ACCTR PIC ZZZZZZZZ.
129 LASTFF-1 PIC XXX.
131 NEW-PAGE-1 PIC X.
132 SETPOS-2A PIC X.
133 MOVHR-2A PIC X.
136 DATE-2 PIC ZZ/ZZ/ZZ.
138 NEWLIN-2B PIC X.
139 FILLER PIC X.
141 NAME PIC X(21).
143 NEWLIN-2C PIC X.
145 HEXF-2C PIC X.
146 STR1 PIC XXX.
148 SETPOS-2D PIC X.
149 MOVHR-2D PIC X.

Figure E-5 (Part 4 of 11). COBOL/400 Program
Figure E-5 (Part 5 of 11). COBOL/400 Program
Figure  E-5 (Part 6 of 11). COBOL/400 Program
223  029200 LINKAGE SECTION.
224  029300 01 DATA-PARM PIC X(256).
225  029400 PROCEDURE DIVISION USING DATA-PARM.

226  029600 THE FOLLOWING DECLARATIVES SECTION IS AN ERROR ROUTINE
227  029900 THAT IS RUN WHEN AN ERROR OCCURS ON THE READ OR WRITE
228  030000 OF THE ICF FILE "K/ICF1ICF". THE ROUTINE MOVES DATA FROM
229  030100 THE I-O FEEDBACK AREA TO THE "ERROR" FORMAT OF THE PRINT
230  030200 FILE "K/ICF1PRT", THE PROGRAM IS TERMINATED RELEASING
231  030300 THE FINANCE DEVICE, AND CLOSING ALL FILES.

232  030400 DECLARATIVES.
233  030700 ERROR-SECTION.
234  030800 USE AFTER EXCEPTION PROCEDURE ON I-O.
235  030900 ERROR-PARAGRAPH.
236  031000 ACCEPT FEEDBACK-DATA FROM FEEDBACK-AREA.
237  031100 MOVE CORRESPONDING FEEDBACK-DATA TO ERRREC-O.
  *  ** CORRESPONDING items for statement 227:
  *  ** FMTNM
  *  ** PGMDEV
  *  ** MAJOR
  *  ** MINOR
  *  ** End of CORRESPONDING items for statement 227
238  031200 WRITE PRT-REC FORMAT IS "ERRREC".
239  031300
240  031400 DROP WS-PGMDEV FROM K/ICF1ICF.
241  031500
242  031600 CLOSE K/ICF1ICF
243  031700 K/ICF1DBF
244  031800 K/ICF1PRT.
245  031900
246  032000 STOP RUN.
247  032100 END DECLARATIVES.
248  032200 INITIALIZE-PROGRAM.
249  032400 OPEN I-O K/ICF1ICF.
250  032500 OPEN I-O K/ICF1DBF.
251  032600 OPEN OUTPUT K/ICF1PRT.
252  032700
253  032900 ACQUIRE WS-PGMDEV FOR K/ICF1ICF.
254  033000
255  033100 ACCEPT EDATE FROM DATE.
256  033200 MOVE ZEROES TO PGM-INDIC-AREA.
  *  ** THE DATA COMES INTO THE PROGRAM FROM THE FINANCE
  *  ** END DECLARATIVES.

Figure E-5 (Part 7 of 11). COBOL/400 Program
STMT SEQNR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7 IDENTFCN S COPYNAME CHG DATE

033700 + CONTROLLER WITH UP TO SEVEN FIELDS. THERE ARE SEVEN FLAGS +
033800 + THAT INDICATE WHICH FIELDS ARE PRESENT: +
033900 + IF FIELD ONE IS PRESENT, FLD1P IS "1", OTHERWISE +
034000 + IT IS " ". +
034100 + IF FIELD TWO IS PRESENT, FLD2P IS "2", OTHERWISE +
034200 + IT IS " ". +
034300 + AND SO ON ... +
034400 + THE DATA IS LOADED SEQUENTIALLY INTO THE AVAILABLE FIELDS +
034500 + SO IF THE OPERATOR ENTERS FIELDS 1, 3, 5, AND 7, THE DATA +
034600 + WILL BE STORED IN INPUT FIELDS 1, 2, 3, AND 4. IT IS +
034700 + NECESSARY TO TEST FOR THE PRESENCE OF THE FIELDS AND MOVE +
034800 + EACH INPUT FIELD INTO THE CORRECT FIELD IN THE PROGRAM. +
034900 + SINCE THIS PROGRAM ONLY DEALS WITH ACCOUNT INQUIRIES, IF +
035000 + AN ACCOUNT NUMBER IS NOT PASSED, A MESSAGE WILL BE SENT. +
035100 +
035200 +-------------------------------------------------------------------
035300

238 035400 + MOVE DATA-PARM TO FNC-INPUT-DATA.
239 035500 + SET FLD TO 1.
240 035600 + IF FLD1P = "1"
241 035700 + THEN
242 035800 + MOVE INPUT-FIELD(FLD) TO ACCTNO
243 035900 + PERFORM PROCESS-TRANSACTION
244 036000 + ELSE
245 036100 + MOVE MSG2 TO TEXT-1
246 036200 + PERFORM BUILD-FORMAT-1.
247 036300 + PERFORM CLEAN-UP.
248 036400 +
249 036500 + PERFORM CLEAN-UP.
250 036600 +
251 036700 +
252 036800 +-------------------------------------------------------------------
253 036900 + BEGIN PROCESSING THE TRANSACTION -
254 037000 + A READ FROM THE DATABASE FILE IS DONE USING THE ACCOUNT +
255 037100 + NUMBER AS THE KEY. IF THE READ WAS SUCCESSFUL, CUSTOMER +
256 037200 + AND ACCOUNT INFORMATION IS SENT BACK TO THE CONTROLLER. +
257 037300 + AND ACCOUNT INFORMATION IS SENT BACK TO THE CONTROLLER. +
258 037400 + IF THE READ WAS UNSUCCESSFUL, A MESSAGE STATING THAT THE +
259 037500 + ACCOUNT NUMBER WAS NOT FOUND WILL BE SENT TO THE +
260 037600 + CONTROLLER. THE FILES ARE THEN CLOSED AND THE PROGRAM +
261 037700 + IS TERMINATED. +
262 037800 +
263 037900 +-------------------------------------------------------------------
264 038000 + PROCESS-TRANSACTION.
265 038100 + MOVE 1 TO ACCOUNT-STATUS.
266 038200 + MOVE ACCTNO TO ACCTNR OF ACCOUNT-REC.
267 038300 + READ K/1DBF INVALID KEY MOVE 0 TO ACCOUNT-STATUS.
268 038400 +
269 038500 + IF VALID-ACCOUNT +
270 038600 + THEN
271 038700 + PERFORM VALID-INQUIRY
272 038800 + ELSE
273 038900 + MOVE MSG1 TO TEXT-1
274 039000 + PERFORM BUILD-FORMAT-1.
275 039100 +

Figure E-5 (Part 8 of 11). COBOL/400 Program
THE PROCEDURE TO DO A VALID INQUIRY SENDS OUT THE DISPLAY IN TWO RECORDS. THE FIRST RECORD HAS THE BASIC CUSTOMER INFORMATION ACCOUNT, NAME, AND ADDRESS. THE SECOND HAS THE BALANCE INFORMATION.

MOVE ACCTNO TO ACCTNR OF FORMATTED-DATA-1.

The customer information is sent to the controller.

END OF CORRESPONDING ITEMS FOR STATEMENT 266.

END OF CORRESPONDING ITEMS FOR STATEMENT 266.
STMT SEQRNR -A 1 2.3.4.5.6.7. IDENTFCN $ COPYNAME CHS DATE
270 043900 MOVE NEKLIN TO NEKLIN-2B, NEKLIN-2C, NEKLIN-2D, NEKLIN-2E.
271 044000 MOVE SEIPDS TO SEIPDS-2A, SEIPDS-2C, SEIPDS-2D, SEIPDS-2E.
272 044100 MOVE MOVHOR TO MOVHOR-2A, MOVHOR-2C, MOVHOR-2D, MOVHOR-2E.
273 044200 MOVE HEXRD TO HEXRD-2A.
274 044300 MOVE HEXDST TO HEXDST-2D, HEXDST-2E.
275 044400 MOVE DTASEP TO DATA-SEP-2.
276 044500 MOVE FFS TO LASTFF-2.
277 044600 MOVE 1 TO MORE-DATA.
278 044900 MOVE FORMATTED-DATA-2 TO FNC-FORMATTED-DATA.
279 045000 MOVE 13H TO LRENCF OF VARREC-0.
280 045200 PERFORM SEND-RESPONSE.
281 045300 045400+:+:-----------------------------------------------
282 045500+:* THE ACCOUNT INFORMATION IS SENT TO THE CONTROLLER.
283 045600+:* +:-----------------------------------------------
284 045700+:* +:-----------------------------------------------
285 045800+:* +:-----------------------------------------------
286 045900 BUILD-FORMAT-3.
287 046000 MOVE CORRESPONDING ACCOUNT TO FORMATTED-DATA-3.
288 046100+:+:-----------------------------------------------
289 046200+:* ++ CORRESPONDING items for statement 281:
290 046300+:* ++ TRAL
291 046400+:* ++ DLYTH
292 046500+:* ++ DLYSEP
293 046600+:* ++ LIMIT1
294 046700+:* ++ LIMIT2
295 046800+:* ++ LCKDSC
296 046900+:* +: End of CORRESPONDING items for statement 281
297 047000 MOVE 0 TO MORE-DATA.
298 047300 MOVE FORMATTED-DATA-3 TO FNC-FORMATTED-DATA.
299 047400 MOVE 20H TO LREC-OF VARREC-0.
300 047600 PERFORM SEND-RESPONSE.
301 047700 047800+:+:-----------------------------------------------
302 047900+:* +: RESPOND TO THE ACCOUNT INQUIRY.
303 048000+:* +:-----------------------------------------------
304 048100+:* +:-----------------------------------------------
305 048200+:* +:-----------------------------------------------
306 048300 SEND-RESPONSE.
307 048400 MOVE FNC-OUTPUT-DATA TO DTAREC OF VARREC-0.
308 048500 Figure E-5 (Part 10 of 11). COBOL/400 Program

STMT SEQRNR -A 1 2.3.4.5.6.7. IDENTFCN $ COPYNAME CHS DATE
296 048600 WRITE ICFCR.
297 048700 FORMAT IS "VARREC".
298 048800 TERMINAL IS WS-PGMDEV.
299 048900 THE ACCOUNT INFORMATION IS SENT TO THE CONTROLLER.
300 049000 BUILD-FORMAT-3.
301 049100 CORRESPONDING items for statement 281:
302 049200 TERM-OF PROGRAM.
303 049300+:* +:-----------------------------------------------
304 049400+:* +:-----------------------------------------------
305 049500 CLEAN-UP.
306 049600 DROP WS-PGMDEV FROM KDO1CF.
307 049700 049800 CLOSE KDO1CF.
308 049900 KDO1DBF.
309 050000 KDOIPRT.
310 050100 STOP RUN.
311 050200 050300+:+ E N D O F S O U R C E +:+
312 050400+:+ E N D O F M E S S A G E S +:+
313 050500+:+ Message Summary
314 050600+:+ Total, Info(0-4), Warning(5-19), Error(20-29), Severe(30-39), Terminal(40-99)
315 050700+:+ 0 0 0 0 0 0 0
316 050800+:+ Source records read . . . . . . . . : 503
317 050900+:+ Copy records read . . . . . . . . : 43
318 051000+:+ Copy members processed . . . . . . : 3
319 051100+:+ Sequence errors . . . . . . . . : 0
320 051200+:+ Highest severity message issued . : 0
321 051300+:+ LBL900 00 Program KDOI created in library FNCLIB.
322 051400+:+ +: E N D O F C O M P I L A T I O N +:+

Figure E-5 (Part 11 of 11). COBOL/400 Program
RPG/400 Source Program for Local System

This section describes the objects needed on the local system to run the ILE RPG/400 CPGEN download finance program. See Figure E-9 on page E-21.

Configuration

This section lists the configuration commands used to create the synchronous data link control (SDLC) line, controller, and device descriptions used by the local system.

CRTLINS DLC LIND(LSYSMON) RSRCNAME(LIN022) ONLINE(+) ROLE(+) NRZI(+) CRTCTL F NC CTLD(CSYSMON) TYPE(4702) MODEL(0) LINKTYPE(+SDLC) ONLINE(+) LINE(LSYSMON) STNADR(1)

CRTDEV F NC DEVD(CPGDEV) TYPE(+FNCICF) LOCADR(01) RMTLOCNAME(CPGDEV) ONLINE(+) CTL(CSYSMON)

Program Files

The following files are used by the local system.

CPGICF

The ICF file used to send and receive records from the finance controller. This file was created by using the following command:

**CRTICFF** FILE(FNCLIB/CPGICF) SRCFILE(FNCLIB/QDDSSRC) SRCMBR(CPGICF)

The following command defines the program device entry:

**ADDICFDEVE** FILE(FNCLIB/CPGICF) PGMDEV(CPGDEV) RMTLOCNAME(CPGDEV)

An OVRICFDEVE command with the same parameters can also be used.

CPGDBF

The database file used to hold the records of the CPGEN file that is to be downloaded. This file was created by using the following command:

**CRTPF** FILE(FNCLIB/CPGDBF) SRCFILE(FNCLIB/QDDSSRC) SRCMBR(CPGDBF)

CPGPRT

The printer file used to format output to a printer. This file was created by using the following command:

**CRTPRT** FILE(FNCLIB/CPGPRT) SRCFILE(FNCLIB/QDDSSRC) SRCMBR(CPGPRT)
Figure E-6. DDS Source for ICF File CPGICF
**Figure E-7. DDS Source for Database File CPGDBF**
Figure E-8. DDS Source for Printer File CPGPRT
Program Explanation: This section explains the ILE RPG/400 CPGEN file download program example in Figure E-9 on page E-21.

1. This section of the program defines the database file (CPGDBF), the ICF file (CPGICF), and the printer file (CPGPRT) used in the program.

CPGDBF is the database file that contains the CPGEN file to be downloaded.

CPGICF is the ICF file that sends records to and receives records from the system monitor program on the finance controller.

CPGPRT is the printer file that sends communications error information to a printer device from the CPGEN file download program on the AS/400 system.

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

2. FEEDBK is the name of the file information data structure (INFDS) used with the CPGICF file. The FEEDBK data structure contains the following information:
   - Record format name (FMTNM)
   - Program device name (PGMDEV)
   - Major/Minor return code (MAJOR, MINOR)

3. WRKBUF is the name of the structure that holds records to be sent to the finance controller and that holds records received from the finance controller.

4. CNSTDS is the name of the structure used by the program for building the initialization and termination sequences that are sent to the system monitor program on the finance controller. The initialization sequence tells the system monitor program that a CPGEN file is coming. The termination sequence tells the system monitor program that the CPGEN file download program on the AS/400 system has finished sending data.

5. FILL00 is the name of the structure used by the program to set records to hex '00'.

6. This section explicitly acquires the program device (PGMDEV) used by the program. This program device was previously added to the ICF file (CPGICF) by the ADDICFDEV command.

7. This section builds and sends the initialization sequence to the finance controller.

8. Data is read from the database file (CPGDBF) and sent to the finance controller until all records in the database file are sent (the indicator 99 is set on).

9. This section of the program sends the termination sequence to the finance controller.

10. The system monitor program on the finance controller sends a response back to the termination sequence sent in 9. If the response received is hex '0581', the CPGEN file download was successful and the indicator 95 is set; otherwise, control is passed to 13 and a message is printed. Then the program is ended.

11. This section of the program does the end-of-job processing. First, the session with the system monitor program on the finance controller is ended by releasing the finance device. Because additional processing is not needed in this program, the LR indicator is set on and all files are closed implicitly. Then the program ends.

12. This subroutine builds the initialization and termination sequences sent to the finance controller and builds constants used throughout the program.

13. When a response is received that indicates the records sent to the system monitor program on the finance controller were rejected, this subroutine prints a printer record (RSPERR) that indicates the CPGEN file download was not successful.

14. This subroutine is automatically called when an exception occurs while the program is running. A printer record (ERRREC) is printed with information regarding the error condition. The *CANCL option on the ENDSR operation causes the program to end and all files to close.
Figure E-9 (Part 1 of 6). RPG/400 Program for ICF Finance
Figure E-9 (Part 2 of 6). RPG/400 Program for ICF Finance
Figure E-9 (Part 3 of 6). RPG/400 Program for ICF Finance
Figure E-9 (Part 4 of 6). RPG/400 Program for ICF Finance
Additional Diagnostic Messages

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOURCE**

**END OF SOU
Figure E-9 (Part 6 of 6). RPG/400 Program for ICF Finance
ILE C/400 Source Program for Local System—Prestarted Job Example

This section describes the objects needed on the local system to run the ILE C/400 account inquiry finance program using a prestarted job example.

Configuration

Use the following configuration commands to create the synchronous data link control (SDLC) line, controller, and device descriptions used by the local system.

```
CRTLINS DLC LIND(FNCLINE) RSRCNAME(LINO22) ONLINE(+NO) 
  ROLE(+PRI) NRZI(+YES) 
CRTCTLFNC CTLD(FNCCCTL) TYPE(4702) MODEL(0) LINKTYPE(+SDLC) 
  ONLINE(+NO) LINE(FNCLINE) STNADR(01) 
CRTDEVFNC DEVD(K/002DEV) TYPE(+FNCICF) LOCADR(03) 
  RMTLOCNAME(K/002DEV) ONLINE(+NO) CTL(FNCCCTL)
```

Use the following configuration commands to add a prestarted job to the subsystem description QCMN and start the subsystem, which automatically starts jobs for the prestarted job entry.

```
ADDPJE SBSD(QCMN) PGM(FNCLIB/K/002) USER(FNCUSER) 
  CLS(FNCCLASS) 
STRSBS SBSD(QCMN)
```

Note: The subsystem description QCMN is IBM-supplied and supports all communications jobs.

Program Files

The following files are used by the local system.

**K002ICF**

The ICF file used to send and receive records from the finance controller. This file was created by using the following command:

```
CRTICFF FILE(FNCLIB/K002ICF) SRCFILE(FNCLIB/QDDSSRC) 
  SRCMBR(K002ICF)
```

The following command defines the program device entry:

```
ADDICFDEV FILE(FNCLIB/K002ICF) PGMDEV(PGMDEV) 
  RMTLOCNAME(+REQUESTER) CMNTYPE(+FINANCE)
```

An OVRICFDEV command with the same parameters can also be used.

**K002DBF**

The database file that holds the account records. This file was created by using the following command:

```
CRTPF FILE(FNCLIB/K002DBF) SRCFILE(FNCLIB/QDDSSRC) 
  SRCMBR(K002DBF)
```

**K002PRT**

The printer file used to format output to a printer. This file was created by using the following command:

```
CRTPRTF FILE(FNCLIB/K002PRT) SRCFILE(FNCLIB/QDDSSRC) 
  SRCMBR(K002PRT)
```
The DDS for the ICF file used in the account withdrawal application program are shown in Figure E-10.

```
SEQNBR *...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...*...
The DDS for the database file used in the account withdrawal application program are shown in Figure E-11.

Figure E-11. DDS Source for Database File K002DBF
The DDS for the printer file used in the account withdrawal application program are shown in Figure E-12.

| File name | K002PRT |
| Library name | FNCLIB |
| Source file containing DDS | QDDSSRC |
| Library name | FNCLIB |
| Source member containing DDS | K002PRT |
| Source member last changed | 01/15/90 9:59:01 |
| Source listing options | +SOURCE +LIST  +NOSELVL |
| DDS generation severity level | 20 |
| DDS flagging severity level | 00 |
| Authority | LIBCRTAUT |
| Replace file | YES |
| Compiler | IBM AS/400 Data Description Processor |

Figure  E-12. DDS Source for Printer File K002PRT

Program Explanation: The following descriptions explain the ILE C/400 account inquiry program example in Figure E-13 on page E-32.

Note: On any type of error (for example, open error or session error), the session ends, a message prints, and the program ends.

1. This section of the program defines the structures used by the files.
2. This section defines the global constants the program uses.
3. This section defines the global variables the program uses. It also defines the common and display/ICF feedback area pointers, and the file pointers.
4. The routines, except the main routine, are prototyped so the compiler knows the type of value returned and the type of parameters passed, if any.
5. The printer file is opened for output, the ICF file is opened for record input/output, and the database file is opened for record input.

K002PRT is the printer file that sends communication information output to a printer device from the account inquiry program on the AS/400 system.

Note: The ICF file is opened with the separate indicator area option specified.

K002ICF is the ICF file that sends records to and receives records from the On-line Terminal Support (OTS) application program running on the finance controller.

K002DBF is the database file that contains the customer account information.

This section handles transaction requests until an unexpected error occurs or the job is ended, in which case the program also
ends. Within the loop, steps 7 through 14 are run.

7 The program device (PGMDEV) used by the program is explicitly acquired. Because this is a prestarted job, the acquire is suspended until a program start request arrives.

Note: This program device was previously added to the ICF file (K002ICF) by the ADDICFDEV command.

8 The parameters passed to the account inquiry program by the finance controller are obtained from the data area used for program initialization parameters (PIP). The account number indicator field and the account number field are the only fields retrieved from the PIP data area.

Note: The PIP data area is created for each prestarted job.

9 This section of the program checks as to whether an account number was received and whether the account number is valid (exists in the database file). If the answer to either of these checks is no, the transaction error flag is set, an appropriate error message is set in the record to be returned to the requesting device, and the error message is issued in step 10.

10 If a transaction error is detected, an error message is sent to the requesting device.

11 If no transaction error is detected, steps 12 through 13 are run.

12 The customer information and account information are copied into the record that is to be sent to the requesting device.

13 A write operation with the account inquiry response is sent to the program device that sent the inquiry request.

Note: For performance reasons, the INVITE keyword is specified as part of DATABUF record format. For more information, see “Responses” on page 7-1.

14 The ICF session is ended when the ICF record format EOSREC, which has the EOS (end-of-session) function specified, is issued.

15 This section determines whether an operation was successful by checking for a major return code of 00. If the operation was successful, a value of 0 is returned; otherwise, a value of 1 is returned.

Note: Because the feedback areas are updated after each ICF file I/O operation, the succ_rc_check function first updates the pointers to the new feedback areas before determining whether the operation was successful.

16 This section does the end-of-job processing. First, a printer record is printed with information regarding the success or failure of the program. If an I/O error occurs, the ICF session is ended by issuing an ICF record format (EOSREC), which has the EOS (end-of-session) function specified. Files used in the program are closed by passing control to step 17. Because no additional processing is needed in this program, the program returns control to the calling environment.

17 This section closes all opened files used in the program. If an error occurs on a close operation, another close is done (which is always successful).
This program handles account inquiry requests from a finance controller. 

NOTE: On any type of error (i.e. open error, session errors...), the session will be terminated, a message printed, and the program will end.

Retrieve various structures/utilities that are used in program.

Figure E-13 (Part 1 of 7). ILE C/400 Program
struct
{
    char ws_type??(2??);  
    char ctl_unit??(2??);  
    char ws_num??(2??);  
    char audit_num??(2??);  
    char lnbmr??(2??);  
    char trans_code??(3??);  
    char resvrd??(3??);  
    char ctl_utype??(1??);  
    char fld1p??(1??); //c5197 Account number field indicator. */  
    char fld2p??(1??);  
    char fld3p??(1??);  
    char fld4p??(1??);  
    char fld5p??(1??);  
    char fld6p??(1??);  
    char fld7p??(1??);  
    char input_fld1??(1/zerodot??); //c5197 Account number. */  
    char input_fld2??(1/zerodot??);  
    char input_fld3??(1/zerodot??);  
    char input_fld4??(1/zerodot??);  
    char input_fld5??(1/zerodot??);  
    char input_fld6??(1/zerodot??);  
    char input_fld7??(1/zerodot??);  
} pip_data;

//c5197 Define the structures used for the data that is to be sent to 
//c5197 the finance controller. These structures are used when doing 
//c5197 "writes" to the ICF file. 

struct
{
    char header1??(16??);  
    char newpage;  
    char newline1;  
    char msg_buf??(4/zerodot??);  
    char newline2;  
    char filler1??(1/zerodot??);  
    char account_num??(1/zerodot??);  
    char end_of_data1;  
    char end_of_data2;  
    char end_of_data3;  
    char end_of_data4;  
} msgrec = { 
    "Account#: ",  
    0xFF, 0xFF, 0xFF, 0xFF };

Figure E-13 (Part 2 of 7). ILE C/400 Program
/* Structure used to send the account information to the controller requesting the transaction. Information that is sent includes: account number, customer name, address, occupation, and the account balance. */

struct
{
    char header1['16'];
    char newline1;
    char filler1['12'];
    char account_num['20'];
    char newline2;
    char filler2['12'];
    char name['15'];
    char newline3;
    char filler3['12'];
    char address['25'];
    char newline4;
    char filler4['12'];
    char city_state_zip['25'];
    char newline5;
    char filler5['12'];
    char occupation['20'];
    char newline6;
    char filler6['12'];
    char balance['20'];
    char end_of_data1;
    char end_of_data2;
    char end_of_data3;
    char end_of_data4;
}
datarec = "000000", 0x0C,
    Bx15, "Account# : "; "",
    Bx15, "Name : "; "",
    Bx15, "Address : "; "",
    Bx15, "City/State: "; "",
    Bx15, "Occupation: "; "",
    Bx15, "Balance : "; "",
    BxFF,BxFF,BxFF,BxFF, };

/* Define structure used to contain the messages that can be sent to the finance controller. */

struct
{
    char msg1['4'];
    char msg2['4'];
}
msg_records = "Account number not received. ", "Account number not found. ";

/* Define structure used to write to the print file. */

struct
{
    char filler1['13'];
    char major['2'];
    char minor['2'];
    char filler2['19'];
}
print_rec;

Figure E-13 (Part 3 of 7). ILE C/400 Program
/*...*/

#define ERROR 1 // Error occurred.
#define NOERROR 0
#define NORM_END 1 // Print normal end message.
#define OPEN_ERR 2 // Print open file error message.
#define ACQ_ERR 3 // Print acquire error message.
#define IO_ERR 4 // Print I/O error message.
#define TRUE 1 // Used in never-ending loop.
#define FALSE 0

#include "fnclib.h" // Include library functions.

int succ_rc_check(void);
void end_job(int);
void close_files(void);

int found = FALSE; // Found account flag.
char dbfkey_z??(1/zerodot??); // Database file key - zoned.
char balance_z; // Account balance - zoned.
int trans_err = NOERROR; // Transaction error indicator.
_SYSindara dsp_indic; // Separate indicator area.

int dtnam[2] = {"PDA", " "};

if (( prtfptr = _Ropen("FNCLIB/K/zerodot/zerodot2PRT", "ar")) == NULL) { printf("Open failed for printer file\n"); exit(ERROR); }
if (( icffptr = _Ropen("FNCLIB/K/zerodot/zerodot2ICF", "ar+, indicators=Y")) == NULL) { end_job(OPEN_ERR); }
if (( dbfptr = _Ropen("FNCLIB/K/zerodot/zerodot2DBF", "rr riofb=n")) == NULL) { end_job(OPEN_ERR); }

Figure E-13 (Part 4 of 7). ILE C/400 Program
The "driver" of this program is contained in the following loop. In the loop, the program:

1. acquires the requester device
2. retrieves the program initiation parameters (PIP data)
3. processes the transaction
4. ends the session

NOTE: The program will terminate if any error occurs.

while ( TRUE )
{
    /* Explicitly acquire the session. */

    if ( succ_rc_check() == ERROR) end_job(ACQ_ERR);

    /* The PIP data that is to be retrieved from the data area */
    /* has seven indicators which indicate which fields are */
    /* present: */
    /* If field one is present, FLD1P is "1", otherwise */
    /* it is ":" 
    /* If field two is present, FLD2P is "2", otherwise */
    /* it is ":" */
    /* and so on . . . */
    /* The data is loaded sequentially into the available fields */
    /* so if the operator enters fields 1, 3, 5, and 7, the data */
    /* will be stored in input fields 1, 2, 3, and 4. It is */
    /* necessary to test for the presence of the fields and move */
    /* each input field into the correct field in the program. */
    /* Since this program only deals with account inquiry requests, */
    /* if an account number is not passed this program will */
    /* reject the transaction request. */
    /* The following code retrieves only those parts of the PIP */
    /* data that this program uses, and rejects the transaction */
    /* request if the account number is not among the data that is */
    /* passed, or if the account number is not in the database file. */

    QXXRTVDA(dtaname, 22, sizeof(pip_data.fld1p), pip_data.fld1p);
    QXXRTVDA(dtaname, 29, sizeof(dbfkey_z), dbfkey_z);
    trans_err = NOERROR;

    if (strncmp(pip_data.fld1p, "1", 1) != 0)
    {
        trans_err = ERROR;
        strncpy(msgrec.msg_buf, msg_records.msg1, 4);
        strncpy(msgrec.account_num, "N/A ", 1);
    }
    else
    {
        strncpy(account_info_rec.account_num, dbfkey_z, 1);
        db_fdbk = _Rreadk(dbfptr, &account_info_rec, sizeof(account_info_rec), __KEY_EQ, &account_info_rec.account_num, sizeof(account_info_rec.account_num));
        if (db_fdbk->num_bytes > 0)
            trans_err = NOERROR;
        else
            trans_err = ERROR;
        strncpy(msgrec.msg_buf, msg_records.msg2, 4);
        strncpy(msgrec.account_num, dbfkey_z, 1);
    }
}

Figure E-13 (Part 5 of 7). ILE C/400 Program
269 ibraries/;                        269
270 /+ message and end the session; otherwise, process the +/ 270
271 /+ transaction and send a response to the transaction 271
272 request. */ 272
273 */ 273
274 11 26 | if (trans_err == ERROR) 274
275 { 275
276 27 | _Rformat(icffptr, "MSGBUF "); 276
277 28 | _Rwrite(icffptr,&msgrec,sizeof(msgrec)); 277
278 29 | if ( succ_rc_check() == ERROR) end_job(IO_ERR); 278
279 } 279
280 else 280
281 { 281
282 31 | strncpy(datarec.account_num, account_info_rec.account_num,1); 282
283 32 | strncpy(datarec.name, account_info_rec.name, 15); 283
284 33 | strncpy(datarec.address, account_info_rec.address, 25); 284
285 34 | strncpy(datarec.city_state_zip, account_info_rec.city_state_zip, 25); 285
286 35 | strncpy(datarec.occupation, account_info_rec.occupation, 1); 286
287 36 | strncpy(datarec.balance, account_info_rec.balance, 1); 287
288 37 | _Rformat(icffptr, "DATABUF "); 288
289 38 | _Rwrite(icffptr,&datarec, sizeof(datarec)); 289
290 39 | if ( succ_rc_check() == ERROR) end_job(IO_ERR); 290
291 } 291
292 | 292
293 41 | _Rformat(icffptr, "EOSREC "); 293
294 42 | _Rwrite(icffptr,NULL, 0); 294
295 | 295
296 } 296
297 | 297
298 9 300 | /******************************************************************************/ 298
299 0 301 | /* INTERNAL FUNCTIONS */ 299
300 1 302 | /******************************************************************************/ 300
301 2 303 | /******************************************************************************/ 301
302 3 304 | /******************************************************************************/ 302
303 4 305 | /******************************************************************************/ 303
304 5 306 | /******************************************************************************/ 304
305 6 307 | /******************************************************************************/ 305
306 7 308 | /******************************************************************************/ 306
307 8 309 | /******************************************************************************/ 307
308 9 310 | /******************************************************************************/ 308
309 0 311 | /******************************************************************************/ 309
310 1 312 | /******************************************************************************/ 310
311 2 | 312
312 { 312
313 [ 313
314 1 | comm_fdbk = _Riofbk(icffptr); 314
315 2 | _Rformat(icffptr, "MSGBUF "); 315
316 3 | if (strncmp(dsp_icf_fdbk->major_ret_code, "00", 2) == 0) 316
317 4 | return(NOERROR); 317
318 5 | else 318
319 6 | return(ERROR); 319
320 7 | } 320
321 | 321

Figure E-13 (Part 6 of 7) ILE C/400 Program
The following function writes message and return code (if any) to a printer file, ends the session with the controller, closes the files used by the program and returns to the caller of this program.

```c
#include <stdio.h>

void end_job(int mtype)
{
    _Rformat(prtfptr, "HEADER ");
    _Rwrite(prtfptr, NULL, 0);
    _Rformat(prtfptr, "PRTREC ");
    if (mtype != OPEN_ERR) {
        strncpy(print_rec.major, dsp_icf_fdbk->major_ret_code, 2);
        strncpy(print_rec.minor, dsp_icf_fdbk->minor_ret_code, 2);
        strncpy(print_rec.filler1, "RETURN CODE: ", 13);
        strncpy(print_rec.filler2, " ", 19);
        _Rwrite(prtfptr,&print_rec, sizeof(print_rec));
    }
    if (mtype == NORM_END) _Rwrite(prtfptr,"PROGRAM COMPLETED NORMALLY ", 36);
    else if (mtype == OPEN_ERR) {
        _Rwrite(prtfptr,"PROGRAM COULD NOT OPEN FILE(S) ", 36);
    }
    else if (mtype == ACQ_ERR) {
        _Rwrite(prtfptr,"PROGRAM COULD NOT ACQUIRE DEVICE ", 36);
    }
    else if ((mtype != OPEN_ERR) && (mtype != ACQ_ERR)) {
        _Rwrite(prtfptr,"I/O ERROR OCCURRED ", 36);
    }
    _Rclose(icffptr);
    _Rclose(prtfptr);
    _Rclose(dbfptr);
    _Rformat(icffptr, "EOSREC ");
    _Rwrite(icffptr,NULL,0);
}
```

Close files:

```c
#include <stdio.h>

void close_files()
{
    if (icffptr != NULL) _Rclose(icffptr);
    if (prtfptr != NULL) _Rclose(prtfptr);
    if (dbfptr != NULL) _Rclose(dbfptr);
}
```

Figure E-13 (Part 7 of 7). ILE C/400 Program
Appendix F. Non-Intersystem Communications Function Finance Example Programs

This appendix provides COBOL/400 and RPG/400 example programs to demonstrate how finance communications is used by using the submit Finance Job (SBMFNCJOB) command.

Figure F-1 shows representations of example programs.

COBOL/400 and RPG/400 Source Programs for the Local System

This section describes the objects needed on the local system to run the ILE COBOL/400 and ILE RPG/400 account inquiry and withdrawal finance example programs.

Configuration

The following configuration commands are used to create the synchronous data link control (SDLC) line, controller, and device descriptions used by the local system.

```
CRTLINS DLC LIND(FNCLINE) RSRCNAME(LIN022) ONLINE(+NO) ROLE(+PRI) NRZI(+YES)
CRTCTLFNC CTLD(FNCTL) TYPE(4702) MODEL(8) LINKTYPE(+SDLC) ONLINE(+NO) LINE(FNCLINE) STNADR(01)
CRTDEVFNC DEVD(FNCDEV) TYPE(4704) LOCADR(03) ONLINE(+NO) CTL(FNCTL)
```

Program File

The following files are used by the local system:

ACCOUNT
The database file that holds the account records.

This file was created by using the following command:

```
CRTPF FILE(FNCLIB/ACCOUNT) SRCFILE(FNCLIB/QDDSSRC) SRCMBR(ACCOUNT)
```

AS/400 System

![AS/400 System Diagram]

Figure F-1. Configuration Example for Non-ICF Finance
The DDS used by this file is shown in Figure F-2.

```
SEQNBR *+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8 Date
 10  A+---------------------------------------------------------------
 20  A  +  DDS  +  
 30  A* +  FOR THE DATABASE FILE  +  
 40  A* +  USE IN ACCOUNT INQUIRY APPLICATION PROGRAM  +  
 50  A* +  
 60  A* +  
 70  A+---------------------------------------------------------------
 80  A* +  
 90  A* +  UNIQUE  

100  A  R  ACCOUNTR  
110  A  ACCTNR  8  0  
120  A  NAME  21  
130  A  STR1  3  
140  A  STR2  18  
150  A  CITY  16  
160  A  ZIP  5  
170  A  OCUF  21  
180  A  TBAL  10  2  
190  A  DLNTH  10  2  
200  A  DLYDEF  10  2  
210  A  LIMET1  10  2  
220  A  LIMET2  10  2  
230  A  RSVCS1  10  2  
240  A  RSVCS2  10  2  
250  A  RSVMM1  10  2  
260  A  RSVMM2  10  2  
270  A  ACTIVE  1  0  
280  A  LKDSDC  38  
290  A  WTHDRL  10  2  
300  A  K  ACCTNR  

++++ END OF SOURCE ++++

Figure F-2 (Part 1 of 2). DDS Source for ACCOUNT File
```

```
SEQNBR *+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8 Date
 10  A+---------------------------------------------------------------
 20  A  +  DDS  +  
 30  A* +  FOR THE DATABASE FILE  +  
 40  A* +  USE IN ACCOUNT INQUIRY APPLICATION PROGRAM  +  
 50  A* +  
 60  A* +  
 70  A+---------------------------------------------------------------
 80  A* +  
 90  A* +  UNIQUE  

100  A  R  ACCOUNTR  
110  A  ACCTNR  8  0  
120  A  NAME  21  
130  A  STR1  3  
140  A  STR2  18  
150  A  CITY  16  
160  A  ZIP  5  
170  A  OCUF  21  
180  A  TBAL  10  2  
190  A  DLNTH  10  2  
200  A  DLYDEF  10  2  
210  A  LIMET1  10  2  
220  A  LIMET2  10  2  
230  A  RSVCS1  10  2  
240  A  RSVCS2  10  2  
250  A  RSVMM1  10  2  
260  A  RSVMM2  10  2  
270  A  ACTIVE  1  0  
280  A  LKDSDC  38  
290  A  WTHDRL  10  2  
300  A  K  ACCTNR  

++++ END OF SOURCE ++++

Figure F-2 (Part 2 of 2). DDS Source for ACCOUNT File
```
This section explains the COBOL/400 account inquiry and withdrawal program example in Figure F-3 on page F-4.

1. This section of the program defines the database file (ACCOUNT).
   ACCOUNT is the database file that contains the customer account information.

2. The program opens the file to be used.

3. The parameters passed to the account inquiry program by the finance controller are placed in a data structure. The account number and the deposit amount fields are the only fields checked in this example. If the account number is not passed by the online terminal support (OTS) program, a message is sent to the finance controller and the program is ended. If a withdrawal amount is passed by OTS, the request is a withdrawal request; otherwise, it is simply an account inquiry request.

4. A read operation from the database file (with the account number as the key) is done.

5. If the account number is not in the database file, a message is sent to the finance controller. If the account number is in the database file, control is passed to either 6 if the OTS request is an account inquiry, or to 7 if the OTS request is a withdrawal transaction.

6. This routine sends out the account inquiry reply in two records. The first record contains customer information, and the second record contains account information.
   **Note:** Because the QFN-write (QFNWRT) routine sends the records to the finance controller, after the records have been sent, the length of data being sent is set to zero. This prevents the finance job from sending any additional data for this transaction to the finance controller.

7. This routine handles withdrawal transactions. If the account is not active, a message is sent to the finance controller and the program is ended. Otherwise, control is passed to 8 (to process the withdrawal transaction).

8. This routine determines whether there are sufficient funds to satisfy the withdrawal request. If not, the withdrawal request is handled as an inquiry request, and a message stating that sufficient funds are not available is included with the inquiry request response. If sufficient funds exist to satisfy the withdrawal request, control is passed to 9.

9. This routine updates the database file to reflect the withdrawal request, and sends the updated account information to the finance controller by using the QFNWRT routine.

10. This routine is called to build and send the error message to the finance controller.
    **Note:** The routine QFNWRT is not used to send the error message. Because the send depth is not zero, the finance job sends the error message automatically to the finance controller when the program ends.

11. This routine is called to build and send the customer information to the finance controller.

12. This routine is called to build and send the account information to the finance controller.

13. This routine is called to build and send the account information to the finance controller when the transaction is a withdrawal request.
The database file is closed and the program is ended.

Program . . . . . . . . . . . . . . : OTSCBL1
Library . . . . . . . . . . . . . : FNCLIB
Source file . . . . . . . . . . . . : PGMSRC
Library . . . . . . . . . . . . . : FNCLIB
Source member . . . . . . . . . . : OTSCBL1
Text 'description' . . . . . . . . : /c5197BLANK
Source listing options . . . . . . : /c5197NONE
Generation options . . . . . . . . : /c5197NONE
Message limit:
    Number of messages . . . . . . . : /c5197NOMAX
    Message limit severity . . . . . : 29
Print file . . . . . . . . . . . . . : QSYSPRT
Library . . . . . . . . . . . . . : /c5197LIBL
FIPS flagging . . . . . . . . . . . : /c5197NOFIPS /c5197NOSEG /c5197NODEB /c5197NOOBSOLETE
SAA flagging . . . . . . . . . . . : /c5197NOFLAG
Flagging severity . . . . . . . . . : 0
Replace program . . . . . . . . . : /c5197YES
Target release . . . . . . . . . . : /c5197CURRENT
User profile . . . . . . . . . . . . : /c5197USER
Authority . . . . . . . . . . . . . : /c5197LIBCRTAUT
Compiler . . . . . . . . . . . . . : IBM AS/400 COBOL/400

Figure F-3 (Part 1 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
Figure F-3 (Part 2 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
Figure F-3 (Part 3 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
Figure  F-3 (Part 4 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
Figure F-3 (Part 5 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
Figure F-3 (Part 6 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
IT IS NECESSARY TO TEST FOR THE PRESENCE OF THE FIELDS AND MOVE
EACH INPUT FIELD INTO THE CORRECT FIELD IN THE PROGRAM.
TO PROCESS THE TRANSACTION FIRST CHECK THE ACCOUNT NUMBER.
TO SEE WHETHER IT WAS ENTERED. IF NO ACCOUNT NUMBER WAS ENTERED,
SEND AN ERROR MESSAGE BACK TO THE OPERATOR.

232 003160  MOVE DATA-PARM TO OTS-INPUT-DATA.
233 003170  SET FLD TO 1.
234 003190  IF FLDIP = '1'
            003200  THEN
235 003210  MOVE INPUT-FIELD(FLD) TO ACCTNO
236 003220  SET FLD TO 2
            09/18/90
            003230  ELSE
237 003240  MOVE MSG(1) TO TEXT-1
238 003250  PERFORM BUILD-FORMAT-1
239 003260  PERFORM CLEAN-UP.
240 003280  IF FLD2P = '2'
            003290  THEN
241 003300  MOVE INPUT-FIELD(FLD) TO AMOUNT-IN
242 003310  MOVE AMOUNT-IN TO AMOUNT
243 003320  COMPUTE AMOUNT = AMOUNT / 100
244 003340  ELSE
            003340  ELSE
245 003350  MOVE ZEROS TO AMOUNT.
246 003360  PERFORM PROCESS-TRANSACTION.
247 003370  PERFORM PROCESS-TRANSACTION.
248 003390  BEGIN PROCESSING THE TRANSACTION -
249 003400  A READ FROM THE DATABASE FILE IS DONE USING THE ACCOUNT
250 003410  NUMBER AS THE KEY. IF THE READ WAS SUCCESSFUL, PERFORM
251 003430  AN ACCOUNT INQUIRY OR A WITHDRAWAL TRANSACTION (DEPENDENT
252 003450  ON WHETHER A WITHDRAWAL AMOUNT WAS PASSED); OTHERWISE, SEND
253 003470  A MESSAGE TO THE FINANCE CONTROLLER STATING THAT THE ACCOUNT
254 003490  NUMBER WAS NOT FOUND.

Figure F-3 (Part 7 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
ONCE THE PROCESSING IS COMPLETE, AND THE PROGRAM IS ENDED, THE FINANCE JOB WILL SEND THE DATA IN DATA-PARM BACK TO THE REQUESTING WORK STATION IF SNDLEN IS GREATER THAN 0. IF SNDLEN IS 0, THE FINANCE JOB WILL SIMPLY SEND AN INVITE TO THE WORK STATION TO ALLOW THE OPERATOR TO INPUT THE NEXT TRANSACTION.

THE PROCEDURE TO DO A VALID INQUIRY SENDS OUT THE DISPLAY IN TWO RECORDS. THE OTS-MODE IS SET TO 1. THIS PUTS THE 4704 DISPLAY INTO LARGE SCREEN MODE WHEREBY 192 CHARACTERS ARE DISPLAYED, THEREBY ALLOWING MORE INFORMATION TO BE DISPLAYED. THE FIRST RECORD HAS THE BASIC CUSTOMER INFORMATION ACCOUNT, NAME, AND ADDRESS. THE SECOND HAS THE BALANCE INFORMATION. BOTH SCREENS ARE SENT USING THE "QFNWRT".

BEFORE PERFORMING A WITHDRAWAL, MAKE SURE THE ACCOUNT IS STILL VALID.
004120 IF IT IS NOT, PERFORM AN INQUIRY INSTEAD AND DISPLAY A * 004130 MESSAGE TO THE OPERATOR INDICATING THE ACCOUNT IS NO LONGER ACTIVE. *
004190 *
004200*******************************************************************************
004220 ATTEMPT-WITHDRAWAL.
004230 IF ACTIVE > 0 THEN
004250 MOVE MSG(3) TO LCKDSC OF ACCOUNT-REC
004260 PERFORM VALID-INQUIRY
004270 ELSE
004280 PERFORM START-WITHDRAWAL.
004290
004300*******************************************************************************
004310 *
004330 BEFORE PERFORMING A WITHDRAWAL, MAKE SURE THE ACCOUNT HAS SUFFICIENT *
004330 FUNDS. IF IT DOES NOT, PERFORM AN INQUIRY INSTEAD AND DISPLAY A *
004340 MESSAGE TO THE OPERATOR INDICATING THERE ARE NOT SUFFICIENT FUNDS *
004350 FOR THE WITHDRAWAL. *
004360 *
004370*******************************************************************************
004390 START-WITHDRAWAL.
004400 COMPUTE WRKBAL = TBAL OF ACCOUNT-REC
004420 + DLYWTH OF ACCOUNT-REC
004430 + LIMIT1 OF ACCOUNT-REC
004440 + LIMIT2 OF ACCOUNT-REC.
004450
004460 COMPUTE TOTRSV = RSVCS1 + RSVCS2 + RSVRM1 + RSVRM2.
004470 COMPUTE TOTAVL = WRKBAL - TOTRSV.
004480
004500 IF AMOUNT > TOTAVL THEN
004510 MOVE MSG(4) TO LCKDSC OF ACCOUNT-REC
004520 PERFORM VALID-INQUIRY
004530 ELSE
004540 PERFORM VALID-WITHDRAWAL.
004550
004600*******************************************************************************
004570 *
004590 THE PROCEDURE TO DO A VALID WITHDRAWAL SENDS OUT ONE RECORD. *
004600*******************************************************************************
004620 VALID-WITHDRAWAL.
004630 ADD AMOUNT TO WTHDRL OF ACCOUNT-REC.
004640 DLYWTH OF ACCOUNT-REC.
004650 COMPUTE TOTAVL = TBAL OF ACCOUNT-REC.
004660 PERFORM BUILD-FORMAT-4.
004670 CALL 'QFNWRT' USING WSID, SNDLEN, DATA-PARM, DATA-TYPE.
004680 REWRITE ACCOUNT-REC.
004690
004700*******************************************************************************
004710 SET SNDLEN BACK TO 0 SO THAT THE FINANCE JOB WILL NOT SEND ANY MORE *
004720*******************************************************************************

Figure  F-3 (Part 9 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
STMT SEQNR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S COPYNAME CHG DATE
004728+ DATA FOR THIS TRANSACTION TO THE FINANCE FACILITY
004728+******************************************
004728+ 004740
280 004750  COMPUTE SNDLEN = 0.
004760
004770+******************************************************************************
004770+ 004780+ AN ERROR MESSAGE WAS SENT TO THE FINANCE CONTROLLER. THE MESSAGE
004780+ WILL BE WRITTEN AUTOMATICALLY BY THE FINANCE JOB UPON
004780+ RETURN FROM THIS PROGRAM.
004780+ 004820+ 004830+******************************************************************************
281 004840+ MOVE ACCTNO TO ACCTNR OF FORMATTED-DATA-1.
282 004850+ MOVE 0 TO OTS-MODE.
283 004860+ MOVE NEWPAG TO NEW-PAGE-1.
284 004880+ MOVE DTASEP TO DATA-SEP-1.
285 004890+ MOVE FFS TO LASTFF-1.
004900
004910+******************************************************************************
004910+ 004920+ SET MORE DATA FLAG TO 0 - THIS IS A COMPLETE RECORD -
004920+ 004930+ AND SET THE SEND LENGTH OF THE FORMATTED DATA + 16.
004940+******************************************************************************
004950
286 004960+ MOVE 0 TO MOREDT.
287 004970+ MOVE FORMATTED-DATA-1 TO OTS-FORMATTED-DATA.
288 004980+ MOVE OTS-OUTPUT-DATA TO DATA-PARM.
289 004990+ COMPUTE SNDLEN = 59.
005000
280 005010+ BUILD-FORMAT-2.
290 005020+ MOVE CORRESPONDING ACCTNR TO FORMATTED-DATA-2.
291 005030+ ** CORRESPONDING items for statement 290:
292 005040+ ** ACCTNR
293 005050+ ** NAME
294 005060+ ** STRI
295 005070+ ** STRZ
296 005080+ ** CITY
297 005090+ ** OCUP
298 005100+ ** End of CORRESPONDING items for statement 290
299 005110+ MOVE EDATE TO DATE-2.
300 005120+ MOVE 1 TO OTS-MODE.
301 005130+ MOVE NEWPAG TO NEW-PAGE-2.
302 005140+ MOVE NEWLIN TO NEWLIN-2B, NEWLIN-2C, NEWLIN-2D, NEWLIN-2E.
303 005150+ MOVE MOVHOR TO MOVHOR-2A, MOVHOR-2C, MOVHOR-2D, MOVHOR-2E.
304 005160+ MOVE HEX2/TO HEX2/2A.
305 005170+ MOVE HEX2/2C, HEX2/2D, HEX2/2E.
306 005180+ MOVE DTASEP TO DATA-SEP-2.
307 005190+ MOVE FFS TO LASTFF-2.
005200
005210+******************************************************************************
005210+ 005220+ SET THE SEND LENGTH TO LENGTH OF THE FORMATTED DATA + 16.
005230+******************************************************************************
308 005240+ MOVE 0 TO MOREDT.

Figure F-3 (Part 10 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance

Appendix F. Non-ICF Finance Example Programs

F-13
Figure F-3 (Part 11 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
Appendix F. Non-ICF Finance Example Programs

Figure F-3 (Part 12 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
Figure F-3 (Part 13 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
Figure  F-3 (Part 14 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance
**RPG/400 Program Explanation**

This section explains the RPG/400 account inquiry and withdrawal program example in Figure F-4 on page F-20.

1. This section of the program defines the database file (ACCOUNT).
   
   **ACCOUNT** is the name of database file that contains the customer account information.

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

2. **OUTPUT** is the name of the structure used with the special file function.

3. **DATA** is the name of the structure that contains the data parameters.

4. **CNSTDS** is the name of the structure that contains the control bytes used for screen control on the 4704.

5. **OTSCTL** is the name of the structure that contains the data to be sent to the controller.

---

**Figure F-3 (Part 15 of 15). COBOL/400 Program OTSCBL1 for Non-ICF Finance**
The finance job passes three parameters to this program: the name of the device that sent the data, the data length, and the data.

This section of the program determines which fields in the data stream were sent by the controller.

This section of the program verifies that the account number exists. If the account number is not found, an error message is sent to the controller.

This section of the program determines the balance available to the customer. If an amount was sent from the controller that is less than the total amount available, a withdrawal is performed on the customer's account. Otherwise, the transaction is an account inquiry transaction.

This section of the program performs the end-of-program processing. The LR indicator is set to on and all files are closed implicitly. Then the program ends.

The output specifications are used together with the special file function. The data is formatted and passed to the program named in the special file function called (SUBEDT). This function then passes the data back during the read operation on file OUTPUT. In this example, the SUBEDT program passes back unchanged data.
Figure F-4 (Part 1 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
Figure F-4 (Part 2 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
**Figure**  F-4 (Part 3 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
SEQUENCE NUMBER 13100 C*
13200 C FLDIP IFEQ ‘2’ B001
13300 C MOVE FLDIP,1 AMNT1 102 001
13400 C ADD I 1 001
13500 C ELSE X001
13600 C Z-ADD AMNT1 001
13700 C END E001
13800 C*
13900 C* IGNORE FIELD 3 IF IT IS PRESENT ... NOT USED IN THIS PROGRAM
14000 C*
14100 C FLDIP IFEQ ‘3’ B001
14200 C ADD I 1 001
14300 C END E001
14400 C******************************************************************************
14500 C** FIRST MAKE SURE THAT AN ACCOUNT NUMBER WAS ENTERED. ▲
14600 C** IF NO ACCOUNT NUMBER OR AN ACCOUNT NUMBER OF ZERO ▲
14700 C** WAS ENTERED, SEND AN ERROR MESSAGE BACK TO THE OPERATOR. ▲
14800 C** ▲
14900 C** IF AN ACCOUNT NUMBER IS PRESENT, CHAIN OUT TO THE ACCOUNT ▲
15000 C** FILE TO GET THE MASTER RECORD. ▲
15100 C** ▲
15200 C** IF THE CHAIN FAILS, SEND AN ERROR MESSAGE BACK TO THE ▲
15300 C** OPERATOR. ▲
15400 C** ▲
15500 C** ▲
15600 C******************************************************************************
15700 C ACCTNR IFEQ =ZERO B001
15800 C MOVE MSG,4 TEXT 20 001
15900 C EXSR OUTP01 001
16000 C ELSE X001
16100 C ACCTNR CHAINACCOUNT 90 1 001
16200 C +IMHO IFEQ ‘1’ B002
16300 C MOVE MSG,1 TEXT 002
16400 C EXSR OUTP01 002
16500 C ELSE X002
16600 C******************************************************************************
16700 C** CHECK TO MAKE SURE THAT THE ACCOUNT IS ACTIVE. ▲
16800 C** IF IT IS NOT, MAKE AMNT1 ZERO...MEANING THAT TRANSACTION ▲
16900 C** IS AN INQUIRY. ALSO SEND A MESSAGE TO THE SCREEN ▲
17000 C** INDICATING THAT THE ACCOUNT IS INACTIVE. ▲
17100 C** ▲
17200 C** ▲
17300 C******************************************************************************
17400 C ACTIVE IFEQ 0 B003
17500 C Z-ADD AMNT1 003
17600 C MOVE MSG,3 TEXT 003
17700 C END E003
17800 C MOVE TRAL BALOUT 102 002
17900 C ADD DYNTH BALOUT 002
18000 C ADD DLYDEP BALOUT 002
18100 C BALOUT ADD LIMIT1 WRKBL 102 002
18200 C ADD LIMIT2 WRKBL 002
18300 C MOVE MSG,5 FELDA 12 002
18400 C ADD RSVCS1 TOTRSV 102 002

Figure F-4 (Part 4 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
Figure F-4 (Part 5 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
Figure F-4 (Part 6 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
Figure F-4 (Part 7 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
SEQUENCE IND DO LAST PAGE PROGRAM
NUMBER *....*....2....*....3....*....4....*....5....*....6....*....7....* USE NUM UPDATE LINE 10

34500 0 NEWLIN
34700 0 'BALANCE'
34900 0 NEWLIN
35000 0 FIELD
35100 0 WRKBALJ + 1
35200 0 NEWLIN
35300 0 NEWLIN
35400 0 TEXT
35500 0 DTASEP 175

B000000 OUTPUT FIELDS FOR RECORD ACCOUNTR FILE ACCOUNT FORMAT ACCOUNTR.
B000001 ACCOUNTNR 5P PACK 8,0
B000002 NAME 26 CHAR 21
B000003 STR1 29 CHAR 3
B000004 STR2 47 CHAR 18
B000005 CITY 63 CHAR 16
B000006 ZIP 68 CHAR 5
B000007 OCUP 89 CHAR 21
B000008 TRAL 95P PACK 10,2
B000009 DLNWTH 101P PACK 10,2
B000010 DLYDEP 107P PACK 10,2
B000011 LIMT1 113P PACK 10,2
B000012 LIMT2 119P PACK 10,2
B000013 RSVCS1 125P PACK 10,2
B000014 RSVCS2 131P PACK 10,2
B000015 RSVRM1 137P PACK 10,2
B000016 RSVRM2 143P PACK 10,2
B000017 ACTIE 149P PACK 1,0
B000018 LCKDSC 182 CHAR 38
B000019 WTHDRL 188P PACK 10,2

**** END OF SOURCE ****

Additional Diagnostic Messages

SEQUENCE IND DO LAST
NUMBER *....*....2....*....3....*....4....*....5....*....6....*....7....*....8

Compile-Time Tables

Table/Array . . . . . . . . . : MSG
35800 ACCOUNT-NR NOT FOUND
35900 INSUFFICIENT FUNDS
36000 ACCOUNT INACTIVE
36100 NO ACCOUNT-NR ENTERED
36200 WORK LIMIT

Table of END Position OFFSETS for FIELDS DESCRIBED USING POSITION NOTATION.

STMT NO POS STMT NO POS STMT NO POS STMT NO POS
27200 1 27300 21 28100 1 28200 21
28300 22 28400 38 28500 39 28600 40
28700 47 28800 62 28900 63 29000 66
29100 80 29200 85 29300 99 29400 100
29500 110 29600 124 29700 125 29800 129
29900 146 30000 152 30100 169 30200 170
30300 182 30400 197 30500 198 30600 236
31400 1 31500 2 31600 3 31700 24
31800 25 31900 28 32000 46 32100 47
32200 52 32300 69 33200 22 33300 23
33400 24 33500 25 33600 33 33700 34
33800 55 33900 56 34000 66 34100 75
34200 76 34300 82 34400 95 34500 96
34600 97 34700 104 34800 124 34900 125
35000 137 35100 152 35200 153 35300 154
35400 174

Figure F-4 (Part 8 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance

Appendix F. Non-ICF Finance Example Programs F-27
Figure F-4 (Part 9 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
Figure F-4 (Part 10 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
Figure F-4 (Part 11 of 11). Source for RPG/400 Program OTSRPG1 for Non-ICF Finance
Figure F-5. ILE RPG/400 Program SUBEDT for Non-ICF Finance
Appendix G. Configuration Examples

This appendix provides configuration examples for Financial Branch System Services (FBSS) finance controllers attached to AS/400 systems using SDLC, token-ring, and X.25 network lines. Each example shows the CL commands used to create the AS/400 configuration objects and the FBSS configuration displays containing values that must match the AS/400 configuration.

Configuration Example for FBSS Controller Using SDLC

The following CL commands can be used to create the AS/400 configuration objects needed to attach an FBSS controller with three attached devices to an AS/400 system over a nonswitched SDLC line.

```
CRTLINSDLC LIND(FBSSLINE) RSRCNAME(LIN071) ROLE(*PRI) CNN(*NONSTPP) NRZI(*YES) DUPLEX(*HALF)
CRTCTLFNC CTLD(CTLFBSS) TYPE(+FBSS) MODEL(0) LINKTYPE(*SDLC) SWITCHED(*NO) LINE(FBSSLINE) EXCHID(05712345) STNADR(C1)
CRTDEVFNC DEVD(FBSSDEV02) TYPE(*FNCICF) LOCADR(02) RMTLOCNAME(DEV02) CTL(CTLFBSS)
CRTDEVFNC DEVD(FBSSDEV03) TYPE(*FNCICF) LOCADR(03) RMTLOCNAME(DEV03) CTL(CTLFBSS)
CRTDEVFNC DEVD(FBSSDEV04) TYPE(*FNCICF) LOCADR(04) RMTLOCNAME(DEV04) CTL(CTLFBSS)
```

Program Explanation

The following displays are used to configure the FBSS controller for SDLC communications with the AS/400 system. Values that must match the AS/400 configuration are described below.

1. The Data Link Control specified on the FBSS Communication Servers display (SDLC) must match the link type (LINKTYPE parameter) specified on the CRTCTLFNC command (see Figure G-1).

2. If an SSCP name is specified for the FBSS controller, the value must match the AS/400 SSCP identifier (SSCPID parameter) specified on the CRTCTLFNC command (see Figure G-2 on page G-2).

3. FBSS Station address must match the station address (STNADR parameter) specified on the CRTCTLFNC command (see Figure G-3 on page G-2).

4. FBSS N.R.Z.I. value must match the value specified for NRZI data encoding (NRZI parameter) on the CRTLINSDLC command (see Figure G-3 on page G-2).

5. FBSS Switched line value must match the value specified for the connection type (CNN parameter) on the CRTLINSIDL command and the SWITCHED parameter on the CRTCTLFNC command (see Figure G-3 on page G-2).

6. Concatenation of the FBSS Identification block and Identification number must match the exchange identifier (EXCHID parameter) specified on the CRTCTLFNC command (see Figure G-3 on page G-2).

7. FBSS Line mode must match the value specified for the DUPEX parameter on the CRTLINSIDL command. If CRTS is specified for the FBSS, DUPEX(*FULL) must be specified on the CRTLINSIDL command (see Figure G-3).

8. Logical unit numbers specified for the FBSS controller must match the local location addresses (LOCADR parameters) specified for the device descriptions (CRTDEVFNC commands) associated with this controller description (see Figure G-4 on page G-2).

---

**Figure G-1. FBSS Communication Servers Display**

<table>
<thead>
<tr>
<th>COMVER</th>
<th>FINANCIAL BRANCH SYSTEM SERVICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC Server</td>
<td>C1</td>
</tr>
<tr>
<td>Data Link Control</td>
<td>SDLC</td>
</tr>
<tr>
<td>Number of Buffers</td>
<td>8</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1997
Program Explanation

The following displays are used to configure the FBSS controller for token-ring network communications with the AS/400 system. Values that must match the AS/400 configuration are described below.

1. The Data Link Control specified on the FBSS Communication Servers display (TRDLC) must match the link type (LINKTYPE parameter) specified on the CRTCTLFNC command (see Figure G-5 on page G-3).

2. If an SSCP name is specified for the FBSS controller, the value must match the AS/400 SSCP identifier (SSCPID parameter) specified on the CRTCTLFNC command (see Figure G-6 on page G-3).

3. FBSS Service access point for PC value must match the destination service access point (DSAP parameter) specified on the CRTCTLFNC command (see Figure G-7 on page G-3).

4. FBSS Service access point for Host/37xx/4700 value must match the source service access point (SSAP parameter) specified on the CRTCTLFNC command (see Figure G-7 on page G-3).

5. FBSS PC address must match the value specified for the adapter address (ADPTADR parameter) on the CRTCTLFNC command (see Figure G-7 on page G-3).

6. FBSS Host/37xx/4700 address must match the value specified for the adapter address (ADPTADR parameter) on the CRTCTLFNC command (see Figure G-7 on page G-3).

7. Concatenation of the FBSS SNA XID Block number and SNA Identification number must match the exchange identifier (EXCHID parameter) specified on the CRTCTLFNC command (see Figure G-7 on page G-3).

8. Logical unit numbers specified for the FBSS controller must match the local location addresses (LOCADR parameters) specified for the device descriptions (CRTDEVFNC commands) associated with

Configuration Example for FBSS Controller Using Token-Ring Network

The following CL commands can be used to create the AS/400 configuration objects needed to attach an FBSS controller with two attached devices to an AS/400 system over a token-ring network line.

- **CRLINTRN** L1ND(FBSSTRN) RSRNAME(LIN0331)
  ADPTADR(400010001234)
- **CRTCTLFNC** CTLD(CFBS5) TYPE+(FBSS) MODEL(0)
  LINKTYPE+(LAN) EXCHID(05711111)
  ADPTADR(400012345000) DSAP(04) SSAP(04)
- **CRTDEVFNC** DEVD(TRNDEV03) TYPE+(FNCIFC) LOCADR(03)
  RMTLOCNAME(TRN03) CTL(CFBS5)
- **CRTDEVFNC** DEVD(TRNDEV04) TYPE+(FNCIFC) LOCADR(04)
  RMTLOCNAME(TRN04) CTL(CFBS5)
this controller description (see Figure G-8 on page G-3).

Figure G-5. FBSS Communication Servers Display

<table>
<thead>
<tr>
<th>COMVER FINANCIAL BRANCH SYSTEM SERVICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Servers \Path ......: C:\FBSSNEW</td>
</tr>
<tr>
<td>PC Server ................... A1</td>
</tr>
<tr>
<td>Data Link Control ........... TRDLC</td>
</tr>
<tr>
<td>Number of buffers .......... 48</td>
</tr>
</tbody>
</table>

Figure G-6. FBSS SSCP Names Display

<table>
<thead>
<tr>
<th>COMVER FINANCIAL BRANCH SYSTEM SERVICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSCP Names \Path ......: C:\FBSSNEW</td>
</tr>
<tr>
<td>PC Server..: A1</td>
</tr>
<tr>
<td>SSCP name/zerodot1 ...... X' '</td>
</tr>
<tr>
<td>SSCP name/zerodot2 ...... X' '</td>
</tr>
<tr>
<td>SSCP name/zerodot3 ...... X' '</td>
</tr>
<tr>
<td>SSCP name/zerodot4 ...... X' '</td>
</tr>
<tr>
<td>SSCP name/zerodot5 ...... X' '</td>
</tr>
<tr>
<td>SSCP name/zerodot6 ...... X' '</td>
</tr>
<tr>
<td>SSCP name/zerodot7 ...... X' '</td>
</tr>
<tr>
<td>SSCP name/zerodot8 ...... X' '</td>
</tr>
<tr>
<td>SSCP name1/zerodot ...... X' '</td>
</tr>
<tr>
<td>SSCP name11 ...... X' '</td>
</tr>
<tr>
<td>SSCP name12 ...... X' '</td>
</tr>
<tr>
<td>SSCP name13 ...... X' '</td>
</tr>
<tr>
<td>SSCP name14 ...... X' '</td>
</tr>
<tr>
<td>SSCP name15 ...... X' '</td>
</tr>
<tr>
<td>SSCP name16 ...... X' '</td>
</tr>
</tbody>
</table>

Figure G-7. FBSS Token-Ring Communications Display

<table>
<thead>
<tr>
<th>COMVER FINANCIAL BRANCH SYSTEM SERVICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Ring Communications \Path ......: C:\FBSSNEW</td>
</tr>
<tr>
<td>PC Server..: A1</td>
</tr>
<tr>
<td>Service access point for PC ......... X'04'</td>
</tr>
<tr>
<td>Service access point for Host/37xx .... X'04'</td>
</tr>
<tr>
<td>PC address ................................ X'12345000'</td>
</tr>
<tr>
<td>Host/37xx address ...................... X'10001234'</td>
</tr>
<tr>
<td>SNA XID Block number ................. X'0577'</td>
</tr>
<tr>
<td>SNA XID Identification number ....... X'11111'</td>
</tr>
</tbody>
</table>

Figure G-8. FBSS Session-Id and LU Assignments Display

Configuration Example for FBSS Controller Using X.25

The following CL commands can be used to create the AS/400 configuration objects needed to attach an FBSS controller with three attached devices to an AS/400 system over an X.25 permanent virtual circuit (PVC).

**CRTLINX**

```
CRTLINX25 LIND(LFBSSX25) RSRCNAME(LIN041) LGLCHL((001 +PVC)) NETADR(00000027) CNNINIT(LOCAL) DFTPKTSIZE(256) MAXPKTSIZE(256) MODULUS(8) DFTWDWSIZE(7)
```

**CRTCTLFNC**

```
CRTCTLFNC CTLD(FBSSIA) TYPE(+FBSS) MODEL(0) LINKTYPE(+X25) SWITCHED(+NO) LINE(LFBSSX25) MAXFRAME(265) EXCHID(05700003) NETLVL(1984) LGLCHLID(001)
```

**CRTDEVFNC**

```
CRTDEVFNC DEVD(DFBSS/02) TYPE(+FNCICF) LOCADR(02) RMTLOCNAME(FBSS02) CTL(FBSSIA) CRTDEVFNC DEVD(DFBSS/03) TYPE(+FNCICF) LOCADR(03) RMTLOCNAME(FBSS03) CTL(FBSSIA) CRTDEVFNC DEVD(DFBSS/04) TYPE(+FNCICF) LOCADR(04) RMTLOCNAME(FBSS04) CTL(FBSSIA)
```

To configure the FBSS controller for X.25 communications, the X.25 link profile must be customized (select X.25 Co-Processor Link Profiles on the Communication Profiles and Emulators display). The FBSS X.25 link profile describes the controller interface to the network; none of the values specified for the X.25 link profile need to be coordinated with AS/400 configuration values when the FBSS and AS/400 system are attached through an X.25 network.
Program Explanation

The following displays are used to complete the FBSS configuration for X.25 communications with the AS/400 system. Several of the values specified on these displays must match the AS/400 configuration. These values are described below.

1. The Data Link Control specified on the FBSS Communication Servers display (X25DLC2) must match the link type (LINKTYPE parameter) specified on the CRTCTLFNC command (see Figure G-9).

2. FBSS Type of circuit value must match the type of logical channel specified by the SWITCHED parameter on the CRTCTLFNC command (see Figure G-11).

3. Concatenation of the FBSS Identification block and Identification number must match the exchange identifier (EXCHID parameter) specified on the CRTCTLFNC command (see Figure G-12).

4. Logical unit numbers specified for the FBSS controller must match the local location addresses (LOCADR parameters) specified for the device descriptions (CRTDEVFNC commands) associated with this controller description (see Figure G-13).

Figure G-9. FBSS Communication Servers Display

Figure G-10. FBSS X.25 Communications Display

Figure G-11. FBSS X.25/SNA Configuration Display

Figure G-12. FBSS X.25/SNA Permanent Circuit Configuration Display

Figure G-13. FBSS Session-Id and LU Assignments Display

Configuration Example for 4702 Controller Using SDLC

Figure G-14 on page G-5 shows a CL program that illustrates the creation of a finance environment consisting of an AS/400 system and a 4702 finance controller with display, finance, and printer devices attached. This example corresponds to the 4702 CPGEN file that is listed after the program.
PGM

I Create the SDLC line description to go to the 47/2 controller

CRTLINSDLC LIN47/202 LSRCNAME(LIN47/201) ONLINE(+NO) ROLE(+PRI) +
     NRZ(+NO)

CRTLINSDLC LIN47/202 LSRCNAME(LIN47/201) ONLINE(+NO) ROLE(+PRI) +
     NRZ(+NO)

I Create the 47/2 controller description

CRTCRTLFC CTLD(CTL47/202) TYPE(47/202) MODEL(+SDLC) +
     ONLINE(+NO) LINE(LIN47/202) STNADR(C1)

CRTDEVDSP DEVD(FNCDSP/209) DEVCLS(+RMT) TYPE(3277) MODEL(+NO) +
     LOCADR(209) ONLINE(+NO) CTL(CTL47/202)

CRTDEVDSP DEVD(FNCDSP/209) DEVCLS(+RMT) TYPE(3277) MODEL(+NO) +
     LOCADR(209) ONLINE(+NO) CTL(CTL47/202)

CRTDEVFNC DEVD(DEVICF) TYPE(+FNCICF) LOCADR(03) RMTLOCNAME(DEVICF) +
     ONLINE(+NO) CTL(CTL47/202)

CRTDEVFNC DEVD(DEV47/204) TYPE(47/204) LOCADR(02) ONLINE(+NO) CTL(CTL47/202)

ENDPGM

Figure G-14. AS/400 Configurations Example for Finance Communications
Program Explanation

The following explains the CPGEN program shown in Figure G-15 and the specific values needed for the parameters for finance communications.

1. Since the ACB parameter is not specified, the SDLC line defaults to NRZI. Therefore, the NRZI parameter on the AS/400 SDLC line description must be specified as NRZI(*YES).

2. The TYPE parameter must be a 4502 application if an SDLC is specified.

3. The LUA parameter on the STATION macro in the CPGEN file must match the LOCADR parameter on the device description.

The APBNM parameter specifies the 4702 application which runs on the specified LU address. The SNA protocol supported for the AS/400 device description at that LU address (local location address) must match the protocol supported by the application. For example, if the 4702 application supports SNA LU type 0, then the AS/400 device must be configured as a finance device.

Figure G-15 is an example of a partial CPGEN that would be used to configure a 4702 finance controller corresponding to Figure G-14 on page G-5.

Figure G-15 (Part 1 of 11). CPGEN Program
Appendix G. Configuration Examples

Figure G-15 (Part 2 of 11). CPGEN Program
PUBLISHT/SYSTEM VIRTUAL VOLUME

PUBLIC/VIRTUAL VOLUME PUBLIC=A, PUBLIC VOLUME RESIDES ON DISK A X
SYSTEM=A, SYSTEM RESET VOLUME IS A DISK A X
PRIVATEPORT=1, VOLUME MANAGER IS ON DCA PORT 1 X
VSTA=14, USE STATION 14 FOR I/O REQUESTS

PRIVATE VOLUME FOR DCA PORT 1

PRIVATE/VIRTUAL VOLUME PRIVATE=A, PRIVATE VOLUME RESIDES ON DISK A X
PRIVATEPORT=1, PRIVATE VOLUME IS FOR DCA PORT 1 X
PRIVATE/VOLUME.PRIVATE1 PRIVATE VIRTUAL VOLUME DATA SET
EJECT

COMLINK MACRO

COMLINK DCL=9600, SPEED OF COMMUNICATION LINK X
TYPE=(45/zerodot2), COMMUNICATION TYPE IS SDLC X
WRAP=N, NO WRAP CAPABILITY ON MODEM X
CNL=26/zerodot, SIZE OF LINK READ BUFFERS X
CNB=24, NUMBER OF LINK READ BUFFERS X
WRT=7, ALLOW MAXIMUM NUMBER WRITES X
CTG=1, X
OPTIONS=(BIND)
EJECT

LOOP MACROS

LOOP ID=1, LOOP ID X
TYPE=L, LOCAL LOOP X
BPS=4800, LOOP SPEED

LOOP ID=2, LOOP ID X
TYPE=L, LOCAL LOOP X
BPS=4800, LOOP SPEED

LOOP ID=3, LOOP ID X
TYPE=L, LOCAL LOOP X
BPS=4800, LOOP SPEED

Figure G-15 (Part 3 of 11). CPGEN Program
Figure G-15 (Part 4 of 11). CPGEN Program
Figure G-15 (Part 5 of 11). CPGEN Program
### SEGSTOR3
**SEGSTOR CLASS=3, CPU COMMUNICATION WORKSTATION**
- SEGSIZE=(2/0, REGSEG - NO EXTRA BYTES)
- 256, WORKSTATION ENVIRONMENT SEGMENT
- 70, ADMIN TRANS HOLD SEGMENT
- 70, TERMINAL INPUT SEGMENT
- 256, TERMINAL OUTPUT & LOG SEGMENT
- 32, DISKETTE INPUT/OUTPUT SEGMENT
- 0, CPU INPUT/OUTPUT SEGMENT
- 0, NOT USED
- 0, USER SEGMENT
- S13=4608

### SEGSTOR4
**SEGSTOR CLASS=4, ADMIN PP PASSTHRU WORKSTATION**
- SEGSIZE=(0, REGSEG - NO EXTRA BYTES)
- 2008, STATION I/O BUFFER SEGMENT
- 60, LOCAL STATION FIELDS SEGMENT
- 60, DEFDMS DATSM MACHINE SEGMENT
- 0, LOCAL ERROR AND DEBUG SEGMENT
- 0, SEGMENT 6 - NOT USED
- 0, SEGMENT 7 - NOT USED
- 0, SEGMENT 8 - NOT USED
- 0, SEGMENT 9 - NOT USED
- 60, ADMIN PP TO ABCS SWAP SEGMENT
- 0, SEGMENT 11 - NOT USED
- 0, SEGMENT 12 - NOT USED
- S13=500

### SEGSTOR5
**SEGSTOR CLASS=5, 470/PC - FILE TRANSFER+PS/PC**
- SEGSIZE=(0, REGSEG - NO EXTRA BYTES)
- 3800, STATION I/O BUFFER SEGMENT
- 60, LOCAL STATION FIELDS SEGMENT
- 60, DEFDMS DATSM MACHINE SEGMENT
- 0, LOCAL ERROR AND DEBUG SEGMENT
- 0, SEGMENT 6 - NOT USED
- 0, SEGMENT 7 - NOT USED
- 0, SEGMENT 8 - NOT USED
- 0, SEGMENT 9 - NOT USED
- 60, ADMIN PP TO ABCS SWAP SEGMENT
- 0, SEGMENT 11 - NOT USED
- 0, SEGMENT 12 - NOT USED
- S13ID=4

---

**Figure G-15 (Part 6 of 11). CPGEN Program**
Figure G-15 (Part 7 of 11). CPGEN Program
STATION 10
STATION ID=10, DEFINE CONTROLLER STATION 10 X
LUA=10, TO COMMUNICATE TO HOST AS 10 X
SS=4, SEGMENT STORAGE IDENTIFIER X
APBNM=(DVSMAP47,3), AP ASSOCIATED WITH THIS WS X
DSM=(J300.24,80,EAB,DCA), DATSM - DCA + EAB X
DSMNUMCK, WITH SPECIAL NUMERIC CHARS X
DELSET=X'04', INITIAL DELIMITER CONTROL X
CPU=Y, HOST COMMUNICATION ALLOWED X
SHARED=N, SHARED WORK STATION OPTION X
STARTUP=Y, CONTROL AT START-UP OPTION X
WRT=1, LCHECK AFTER ONE LWRITE CP X
INTMR=6, ALLOW SIX INTERVAL TIMERS X
INSTR=65535, ALLOW MAXIMUM BEFORE LEXIT X
RETSK=12 ALLOW 12 LINK STACK LEVELS X

STATION 10
STATION ID=10, DEFINE CONTROLLER STATION 10 X
LUA=10, TO COMMUNICATE TO HOST AS 10 X
SS=4, SEGMENT STORAGE IDENTIFIER X
APBNM=(DVSMAP47,3), AP ASSOCIATED WITH THIS WS X
DSM=(J300.24,80,EAB,DCA), DATSM - DCA + EAB X
DSMNUMCK, WITH SPECIAL NUMERIC CHARS X
DELSET=X'04', INITIAL DELIMITER CONTROL X
CPU=Y, HOST COMMUNICATION ALLOWED X
SHARED=N, SHARED WORK STATION OPTION X
STARTUP=Y, CONTROL AT START-UP OPTION X
WRT=1, LCHECK AFTER ONE LWRITE CP X
INTMR=6, ALLOW SIX INTERVAL TIMERS X
INSTR=65535, ALLOW MAXIMUM BEFORE LEXIT X
RETSK=12 ALLOW 12 LINK STACK LEVELS X

SPACE

******************************************************************************
*                           DATSM NUMERIC CHECK MACRO                      *
*                                                                         *
******************************************************************************
SPACE
DSMNUMCK DSMNUMCK FC, ALLOW DUP KEY WITHIN NUMERIC X
FE ALLOW FIELD MARK WITHIN NUMERIC X
EJECT

******************************************************************************
*                          DEFADDR AND DCAPORT MACROS                      *
*                                                                         *
******************************************************************************
SPACE
LIA1 DEFADDR (DS470477), ADDRESS 12 77 KEY KEYBOARD X
(2P471001,4), ADDRESS 01 ADDRESS SHARED X
ADDR=(L1,1,4), LOOP 1 ADDRESS 1 120 CPS X
SPACE
LIA2 DEFADDR (DS470450), ADDRESS 11 50 KEY KEYBOARD X
(2P471001,4), ADDRESS 01 ADDRESS SHARED X
ADDR=(L1,1,4), LOOP 1 ADDRESS 4 120 CPS X
SPACE

Figure G-15 (Part 8 of 11). CPGEN Program
L1A3 DEFADDR (JP472/04,4,POOL1), 472 MODEL 04 - DPOOLED PRINTER X
   ADDR=(L1,3,4) LOOP 1 ADDRESS 3 @120 CPS

L1A4 DEFADDR (JP472/04,4,POOL1), 472 MODEL 04 - DPOOLED PRINTER X
   ADDR=(L1,4,4) LOOP 1 ADDRESS 4 @120 CPS

L2A1 DEFADDR (L2A1DBG), 472/04 MODEL 11 & 62 KEYBOARD X
   ADDR=(L2,1,2) LOOP 2 ADDRESS 1 @240 CPS

DC0 DCAPORT (DCO4704),
   PORT=12 DCA PORT ADDRESS 12

DC1 DCAPORT (DC14740),
   PORT=12 DCA PORT ADDRESS 1

DC3 DCAPORT (DC33279),
   PORT=3 DCA PORT ADDRESS 3

DC7 DCAPORT (DC7UTILITY),
   PORT=7 DCA PORT ADDRESS 7

EJECT

******************************************************************************
* ^ DISPLAY DEVXXXX MACROS *
* ^
******************************************************************************

SPACE

DS470408 DEF4704 MODE=11, MODEL 11 KEYBOARD DISPLAY X
   OUTRTBL=OUT4704, OUTPUT TRANSLATE TABLE X
   CHARSET=US, CHARACTER SET FOR U.S. X
   TRTBHDR=KB1ABC50, 50 KEY TRANSLATE TABLE X
   OPTIONS=(49/5,NAME,SL), MAGNETIC STRIPE DEVICE X
   MSTRTBL=(MSABCIN,MSABCST), MAGNETIC STRIPE TABLES X
   ALITE=(SYS), ALARM LIGHTS X
   EOMSET='FF', INITIAL EOM MASK SETTING X
   CURSOR=Y, CURSOR TO STAY ON SCREEN X
   ERTLS=Y LIMIT THE KEYBOARD LREAD

SPACE

DS470477 DEF4704 MODE=12, MODEL 12 KEYBOARD DISPLAY X
   OUTRTBL=OUT4704, OUTPUT TRANSLATE TABLE X
   CHARSET=US, CHARACTER SET FOR U.S. X
   TRTBHDR=KB1ABC57, 57 KEY TRANSLATE TABLE X
   OPTIONS=(49/5,NAME,SL), MAGNETIC STRIPE DEVICE X
   MSTRTBL=(MSABCIN,MSABCST), MAGNETIC STRIPE TABLES X
   ALITE=(SYS), ALARM LIGHTS X
   EOMSET='FF', INITIAL EOM MASK SETTING X
   CURSOR=Y, CURSOR TO STAY ON SCREEN X
   ERTLS=Y LIMIT THE KEYBOARD LREAD

SPACE
Bibliography

The following AS/400 books contain information you may need. The 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 you use this guide:

- **ICF Programming**, SC41-5442, contains information about writing application programs that use inter-system communications function (ICF).
- **Communications Management**, SC41-5406, provides communications support information for the AS/400 system. This includes management information, communications status and errors, and work management.
- **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.
- **Remote Work Station Support**, SC41-5402, contains information and examples on how to configure your system to attach to and use remote work stations and facilities. Also, this guide contains information about the display station pass-through function.
- **DDS Reference**, SC41-5712, contains information about coding data descriptions specifications for physical, logical, display, printer, and ICF files.
- **Data Management**, SC41-5710, provides the application programmer with information about using data management support, which allows an application to work with files.
- **CL Programming**, SC41-5721, provides a wide-range discussion of AS/400 programming topics.
- **CL Reference**, contains information on control language commands.
- **Work Management**, SC41-5306, contains information on how to create an initial management environment and how to change a work management environment.
- **Security – Reference**, SC41-5302, provides information on resource security.

Programming Language Books

The following books contain information about finance communications-supported programming languages:

- **Languages: Systems Application Architecture**
  - **AD/Cycle**
  - **COBOL/400**
  - Reference, SC09-1380
- **Languages: Systems Application Architecture**
  - **AD/Cycle**
  - **COBOL/400**
  - Reference Summary, SX09-1209
- **Languages: Systems Application Architecture**
  - **AD/Cycle**
  - **COBOL/400**
  - User’s Guide, SC09-1383
- **Languages: Systems Application Architecture**
  - **AD/Cycle**
  - **RPG/400**
  - Reference, SC09-1349
- **Languages: Systems Application Architecture**
  - **AD/Cycle**
  - **RPG/400**
  - User’s Guide, SC09-1348
- **Languages: Systems Application Architecture**
  - **C/400**
  - Reference Summary, SX09-1217
- **Languages: Systems Application Architecture**
  - **C/400**
  - User’s Guide, SC09-1347

Personal Banking Machine Books

The following books contain information regarding the operation and problem determination of the personal banking machines:

- **IBM 4730 Personal Banking Machine Series**
  - **Customization Image Builder General Information**, GC31-0029
- **IBM 4730 Personal Banking Machine Series**
  - **Network Monitor General Information**, GC31-0033
- **IBM 4731 Personal Banking Machine General Information**, GA19-5346
- **IBM 4731 Personal Banking Machine Operator’s Quick Reference**, GX11-6098
- **IBM 4731, 4732, and 4736 Personal Banking Machines Customization Guide**, GA19-5353
- **IBM 4731, 4732, and 4736 Personal Banking Machines Error Log Reference Guide**, GA19-5379
- **IBM 4732 Personal Banking Machine General Information**, GA34-2017
Financial Branch System Services (FBSS) Books
The following guides contain information regarding Financial Branch System Services (FBSS):

- IBM Financial Branch System Services Version 2.2, Application Programming, SC19-5174

Systems Network Architecture (SNA) Books
The following books contain information regarding Systems Network Architecture (SNA):

- Systems Network Architecture Reference Summary, GA27-3136

Miscellaneous Books
The following books contain additional information you may need when you use this guide:

- Check Processing Executive/VS: Program Logic Manual, LY20-2556
- Check Processing Executive/VS: Program Reference and Operations, SH20-2496
- Check Processing Executive/3694: Program Logic Manual, LY20-2525
- Check Processing Executive/3694: Program Reference and Operations, SH20-2495
- Host Support User’s Guide, SC31-0020
- IBM 4700 Finance Communications System Controller Programming Library:
  - Communications Programming, GC31-2068
  - Control Program Generation, GC31-2071
  - Cryptographic Programming, GC31-2070
  - Disk and Diskette Programming, GC31-2067
  - General Controller Programming, GC31-2066
  - Work Station Programming, GC31-2069
- IBM 4700 Finance Communications System: Subsystem Operating Procedures, GC31-2032
- IBM 4700 Finance Communications System: Subsystem Problem Determination Guide, GC31-2033
- IBM 4700 Finance Communications System: System Monitor Guide and Reference, GA34-2108
- IBM 4704 Display Station Operating Instructions, GC31-2025
Index

Special Characters
*FNCICF 2-3, 3-2
*REQUESTER value, RMTLOCNAME parameter 5-2

Numerics
3270 emulation 3-3
3601 finance controller 2-1
3694 document processor controller 7-7
3694 document processor, starting 5-4
4701 finance controller 2-1
4701 finance controller, diskette download D-1
4702 CPGEN file G-4
4702 finance controller 2-1
4731 Personal Banking Machine 2-1
4732 Personal Banking Machine 2-1
4736 Self-Service Transaction Station 2-1
47xx controller, starting 5-3
843567, RPQ 2-2
843568, RPQ 2-2

A
account inquiry example F-1
acquire operation 5-4
Add Communications Entry (ADDCMNE) command 4-2
Add ICF Device Entry (ADDICFDEVE) command 5-2
adding finance support user IDs 3-6
application programs communications with 1-1
considerations 7-6
errors 5-7
interface 2-3
using acquire operation 5-4
writing ICF finance 5-1
writing non-ICF finance 6-1
AS/400 books H-1
AS/400 finance communications configuring 3-1
ASCVRFOFF parameter 4-2

B
BIND command 5-2, 5-8
books
AS/400 systems H-1
financial branch system services (FBSS) H-2
miscellaneous H-2
personal banking machines H-1
programming language H-1

books (continued)
Systems Network Architecture (SNA) H-2
buffering considerations 7-1
bytes 6-1

C
C Set ++ for OS/400 1-1
cancel function 5-8
cancel-invite function 5-8
CFGOBJ parameter 4-1
CFGTYPE parameter 4-1
Change Communications Entry (CHGCMNE) command 4-2
Change Controller Description (Finance) (CHGCTLFNC) command 3-2
Change Device Description (Finance) (CHGDEVFNC) command 3-2
Change ICF Device Entry (CHGICFDEVE) command 5-2
Change ICF File (CHGICFF) command 5-1
Change Job Description (CHGJOB) command 7-5
Change Line Description (Ethernet) (CHGLINETH) command 3-1
Change Line Description (SDLC) (CHGLINSDL) command 3-1
Change Line Description (Token-ring) (CHGLINTKN) command 3-1
Change Line Description (X.25) (CHGLINX25) command 3-1
close operation 5-9
COBOL/400 language E-1
commands
ADDCMNE 4-2
ADDICFDEVE 5-2
BIND 5-2, 7-8
CHGCMNE 4-2
CHGCTLFNC 3-2
CHGDEVFNC 3-2
CHGICFDEVE 5-2
CHGICFF 5-1
CHGJOB 7-5
CHGLINETH 3-1
CHGLINSDL 3-1
CHGLINTKN 3-1
CHGLINX25 3-1
CRTCTLFNC 3-2
CRTDEVDS 3-2
CRTDEVFNC 3-2
CRTDEVPRT 3-2
CRTDSPF 6-14
CRTICFF 5-1
commands (continued)
CRTLINETH 3-1
CRTLINS DLC 3-1
CRTLINTRN 3-1
CRTLINX25 3-1
CRTSBSD 4-2
DLTF 5-1
DLTLIND 3-1
DLTOVR 5-1
DLTOVRDEVE 5-2
DSPFD 5-1
DSPFFD 5-1
DSPJOB 7-5
DSPJOBLOG 6-3
DSPLIND 3-1
DSPOVR 5-2
GRTOBJAUT 7-5
LUSTAT 6-18
OVRICFDEVE 5-2
OVRICFF 5-1
RMVC MNE 4-2
RMVICFDEVE 5-2
SBMFNCJOB 6-1
SNA 5-2
SNDFNCIMG D-1
VRYCFG 4-1
WRKACTJOB 6-3
WRKCFS GS 4-1
WRKDEVTBL 3-5
WRKJOBQ 6-3
WRKPGMTBL 3-7
WRKUSR TBL 3-6
communications
configurations, associating program device names 5-2
finance 6-1
finance program, starting 5-2
writing ICF finance application programs 5-1
writing non-ICF application programs 6-1
concurrent sessions 2-2
configuration
configuring controllers 2-1
device description 3-2
finance support 3-1
line description 3-1
non-ICF finance, using Work with Table commands 3-5
objects 4-1
program device names to communications 5-2
types 4-1
examples
FBSS controller using SDLC G-1
FBSS controller using token-ring network G-2
FBSS controller using X.25 PVC G-3
finance communications G-4
SDLC configuration for FBSS controller G-1
consumer transaction facilities (CTF) 3-2
control bytes 6-15
controller
applications 7-6
configuring 2-1
description 3-1
diskette download, 4701 finance D-1
finance session initiation 7-7
number of devices allowed 3-3
processing transaction T001 6-6
program generator (CPGEN) file 2-2
types 7-7
used with ICF and non-ICF finance 2-1
controlling subsystem 4-2
CPGEN file 2-2, G-4
Create Controller Description (Finance)
(CRTCTL FNC) command 3-2
Create Device Description (Display) (CRTD VDSP) command 3-2
Create Device Description (Finance) (CR TDEV FNC) command 3-2
Create Device Description (Printer) (CRTDE VPR T) command 3-2
Create Display (CRTDS PF) command 6-14
Create ICF File (CRTC F) command 5-1
Create Line (Ethernet) (CRTLIN ETH) command 3-1
Create Line Description (CRTLINSDLC) command 3-1
Create Line Description (CRTLINX25) command 3-1
Create Line Description (Token-ring) (CRTLINTRN) command 3-1
Create Subsystem Description (CRTSBSD) command 4-2
CTF (consumer transaction facilities) 3-2

D
data
flow examples 6-4
receiving 5-5
sending 5-4
streams 3-3
data description specifications (DDS)
considerations A-3
defining record formats 5-1
supported keywords A-3
DATA parameter 6-8
DDS (data description specifications)
considerations A-3
defining record formats 5-1
supported keywords A-3
default communications entries 4-2
default entries 4-2
defining devices 3-3
Delete File (DLTF) command 5-1
Delete Line Description (DLTLIND) command 3-1
Delete Override (DLTOVR) command 5-1
Delete Override Device Entry (DLTOVRDEVE) command 5-2
deleting finance support user IDs 3-6
destination logical unit (DLU) 5-4
DEVD parameter 3-7
device
descriptions 2-3, 3-2
tables 3-5
using 3270 3-3
DEVBLT parameter 6-2
diskette download support 2-3, D-1
Display File Description (DSPFD) command 5-1
Display File Field Description (DSPFFD) command 5-1
Display Job Description (DSPJOBD) command 7-5
Display Job Log (DSPJOBLOG) command 6-3
Display Line Description (DSPFLIND) command 3-1
Display Override (DSPOVR) command 5-2
DLU (destination logical unit) 5-4
document processor controller, 3694 7-7
downloading 4701 finance controller diskette 2-3, D-1
examples (continued)
ILE COBOL/400 source program (continued)
DDS source for printer file E-3
ICF file E-2
ILE RPG/400 Call to QFNREAD 6-9
ILE RPG/400 Call to QFNWRTI 6-9
ILE RPG/400 source program
Database file E-17
DDS source for printer file E-18
ICF file E-16
ILE RPG/400 Program SUBEDT for Non-ICF
Finance F-30
RPG/400 program E-20
RPG/400 Call to QFNREADI 6-10
RPG/400 source program
RPG/400 Program OTSRPG1 for Non-ICF
Finance F-19
SDLC configuration for FBSS controller G-1
token-ring network configuration for FBSS controller G-2
X.25 PVC configuration for FBSS controller G-3
exchange identifier (EXCHID) 2-1
fail functions 5-7
FBSS (financial branch system services)
books H-2
considerations 1-1
controller 2-1
starting 5-3
using a token-ring 2-1
using Ethernet 2-1
using SNA 2-1
feedback area 5-10
finance
application programs, writing non-ICF finance 6-1
controller session initiation 7-7
DDS keywords A-3
device table 3-5
ICF device description (*FNCICF) 3-2
non-ICF interface capabilities 6-1
starting a session 5-3
starting subsystem 4-2
support 4-1
table of system-supplied formats A-3
tables A-3
types 3-2
finance communications
ending a session 3-1, 5-8
environment 1-1
functions supported 2-1
interfaces 2-3
introduction 1-1
network example 1-2
overview 2-1
Index X-3
finance communications (continued)

performance 7-1
programming 7-1
return codes B-1
running 4-1
security 2-2, 7-2
SNA 7-1
starting a session 5-2
  table 1-1
  using SNA pass-through 1-2
  varying on and off 4-1

finance input/output manager (FIOM)

error handling 6-10
  routines 6-8
  using SBMFNCJOB command 6-7, 6-11
  using with SBMFNCJOB command 2-3
  using without SBMFNCJOB command 6-13

financial branch system services (FBSS)

books H-2
  considerations 1-1
  controller 2-1
  starting 5-3
  using a token-ring 2-1
  using Ethernet 2-1
  using SNA 2-1

FIOM (finance input/output manager)

error handling 6-10
  routines 6-8
  using SBMFNCJOB command 6-7, 6-11
  using with SBMFNCJOB command 2-3
  using without SBMFNCJOB command 6-13

FMTSLT parameter 5-2
force-data function 5-5
formats, system-supplied A-3
function-management-header function 5-5
functions
  cancel 5-8
  cancel-invite 5-8
  end-of-group (ENDGRP) 5-5
  end-of-session (EOS) 5-9
  fail 5-7
  force-data 5-5
  function-management-header 5-5
  invite 5-6
  negative-response 5-8
  timer 5-8
functions supported by finance communications 2-1

G

get-attributes operation 5-8
Grant Object Authority (GRTOBJAUT) command 7-5

H

high-level file status 5-10
high-level languages (HLL) supported 1-1
Host Diskette Image Create (HDIC) program D-2

I

I/O (input/output)
  error handling 6-3
  feedback area 5-10

ICF
  See intersystem communications function
ILE C/400 language 1-1
ILE COBOL/400 language 1-1, F-1
ILE RPG/400 language 1-1, F-1
indicators 5-9

INIT-SELF
  command 5-3
  command field format 7-7
  data stream 6-16
  request 6-16
  response to 6-17
initial program load (IPL) 4-1
input/output (I/O)
  error handling 6-3
  feedback area 5-10
interfaces for finance 2-3

intersystem communications function (ICF)
  considerations 7-1
  finance
  controllers used with 2-1
  device description (*FNCICF) 3-2
  example programs E-1, F-1
  file commands 5-1
  SNA considerations 7-7
  starting a session 5-2
  starting subsystem 4-3
  support 2-3
invite function 5-6
IPL (initial program load) 4-1

J

job log 2-2
JOB NAME parameter 6-2
JOB parameter 6-3
JOBD parameter 6-2
jobs
  controlling finance devices 6-4
  using program start requests, prestart 7-2

L

language operations, table A-1
LOCADR parameter 2-3, 3-2
Index

logical unit address (LU) 3-2
Logical Unit Status (LUSTAT) command 6-18
LU (logical unit address) 3-2

Override ICF File (OVRICFF) command 5-1
overview of finance communications 2-1

P
parameters
  ASCVRYOFF 4-2
  CFGOBJ 4-1
  CFGTYPE 4-1
  DATA 6-8
  DEVD 3-7
  DEVTBL 6-2
  DFTWAIT 7-4
  FMTSLT 5-2
  JOB 6-3
  JOB NAME 6-2
  JOBID 7-4
  LOCADR 3-2
  MAXLENRU 5-5
  MAXRCDLEN 5-6
  MSGQ 6-2
  PGMDEV 5-2
  PGMTBL 6-2
  program initialization 7-2
  RANGE 4-1
  RMTLOCNAME 5-2
  STATUS 4-1
  TYPE 2-3, 3-2
  USER 6-2
  USRTBL 6-2
  VRYWAIT 4-1
  WAITFILE 5-4

personal banking machine books H-1
personal banking machines 2-1
PGMDEV parameter 5-2
PGMTBL parameter 6-2
prestart jobs using program start requests 7-2
problem notification 5-7
processors 1-1
program initialization parameters 7-2
program start request
errors B-1
prestart jobs using 7-2
syntax 5-3

program tables 3-7
programming language books H-1
programs
  examples of E-1, F-1, G-1
  writing ICF 5-1
  writing non-ICF 6-1

Q
QBASE subsystem descriptions 4-2
QCMN subsystem descriptions 4-2

M
magnetic-ink character recognition (MICR) 2-1
major and minor return codes 5-10, B-1
mapping ICF finance and Systems Network Architecture C-1
MAXLENRU (maximum length of request/response unit) parameter 5-5
MAXRCDLEN parameter 5-6
messages B-1
MICR (magnetic-ink character recognition) 2-1
miscellaneous books H-2
MSGQ parameter 6-2
multiple response indicators 5-10
multipoint line 2-2

N
negative-response function 5-8
non-input/output error handling 6-3
non-intersystem communications function
  BIND command field format 7-8
  finance
    controllers used with 2-1
    interface capabilities table 6-1
    SNA considerations 7-8
    starting subsystem 4-2
    using Work with Table commands 3-5
    writing applications programs 6-1
  programming considerations 7-2
  support 2-3
  notification of problems 5-7

O
object configuration 4-1
open/acquire operation 5-4
operational diskette D-1
operations
  close 5-9
  get-attributes 5-8
  read 5-5
  read-from-invited-program-devices 5-6
  release-attributes 5-9
  waiting for
    data queue 5-6
    display file 5-6
    ICF file 5-6
  write 5-5
Override ICF Device Entry (OVRICFDEVE) command 5-2

notification of problems 5-7
QFN-read (QFNREAD) routine 6-9
QFN-read-invited (QFNREADI) routine 6-10
QFN-write (QFNWRT) routine 6-8
QFN-write-invite (QFNWRTI) routine 6-8
QFN subsystem, using 6-2
QFNDEVTBL file 7-6
QFNPGMTBL file 7-6
QFNUSRBL file 7-6
QLLC (qualified logical link control) 5-5
qualified logical link control (QLLC) 5-5

R
RANGE parameter 4-1
RCVENDGRP response indicator 5-10
RCVFHM response indicator 5-10
RCVNEG response indicator 5-10
read operation 5-5
read-from-invited-program-devices operation 5-6
receive end-of-group (RCVENDGRP) response indicator 5-10
receive function-management-header (RCVFHM) response indicator 5-10
receive negative-response (RCVNEG) indicator 5-10
receiving data 5-5
recovery, error handling support 2-2
related printed information H-1
release operation 5-9
remote location name 5-2
Remove Communications Entry (RMVCNE) command 4-2
Remove ICF Device Entry (RMVICFDEVE) command 5-2
request
prestart jobs using program start 7-2
syntax, program start 5-3
response 7-1
response indicators
definition 5-9
receive-end-of-group 5-10
receive-function-management-header 5-10
receive-negative-response 5-10
return codes
description 5-10
detailed descriptions B-1
examples 5-10
handling errors 2-2
major and minor 5-10
RMTLOCNAME parameter 5-2
RPG/400 language E-1
RPQ 2-2

S
SDLC (synchronous data link control)
configuration for FBSS controller G-1
configurations 2-2
example program E-1, F-1, G-1
finance communications 3-1
frame size error B-12
security 2-2, 7-2
self-service transaction station 2-1
Send Finance Diskette Image (SENDENCIMG)
call D-1
sending data 5-4
sense codes B-1
sessions
communications finance, starting 5-2
concurrent 2-2
initiation, successful finance controller 7-7
starting 5-2
SNA LU (Systems Network Architecture logical unit) 1-1
SNA pass-through
description 1-2
SNA pass-through communications configuring 3-4
start request errors B-27
starting
finance program 5-3
finance subsystem 4-2
session
3694 Document Processor 5-4
47xx controller 5-3
FBSS controller 5-3
STATUS parameter 4-1
Submit Finance Job (SBMFNCJOB) command 7-3
subsystem
QBASE 4-2
QCMN 4-2
QFNC 6-2
starting 4-2
subsystem, starting finance 4-2
supervising finance jobs 6-3
support
concurrent session 2-2
configuring 3-1
error handling and recovery 2-2
for finance communications 1-1
recovery 2-2
running finance communications 4-1
security, finance communications 2-2
using data description specifications (DDS)
keyword A-3
using diskette download 2-3
synchronous data link control (SDLC)
configuration for FBSS controller G-1
configurations 2-2

synchronous data link control (SDLC) (continued)
example program E-1, F-1, G-1
finance communications 3-1
frame size error B-12
syntax, program start request 5-3
system monitor explanation 2-3
system-supplied formats A-3
System/370 computer D-1
Systems Network Architecture (SNA)
books H-2
commands 5-2
considerations 7-7, 7-8
logical unit (SNA LU) 1-1
mapping to C-1
token-ring network 2-1

T

tables
DDS keywords, finance A-3
interface capabilities, non-ICF finance 6-1
language operations A-1
overview of finance communications 1-1
supported DDS keywords A-1
system-supplied formats A-1, A-3
work with program table command 3-7
work with table commands 3-5
TERM-SELF
data stream 6-17
response to 6-18
timer function 5-8
token-ring network 2-1
TYPE parameter 2-3
types of finance devices 3-2
types, configuration 4-1

U

UDDS (user-defined data stream) 6-14, 7-6
UNBIND 5-3
user IDs, adding 3-6
USER parameter 6-2
user tables 3-6
user-defined (USRDFN) keyword 6-14
user-defined data stream (UDDS) 6-14, 7-6
USRDFN keyword 6-14
USRTBL parameter 6-2

V
V.24 2-1
V.35 2-1
Vary Configuration (VRYCFG) command 4-1
varying finance, on and off 4-1
VRYWAIT parameter 4-1

W
WAITFILE parameter 5-4
waiting for
data queue 5-6
DATQ parameter commands 5-6
description 5-6
display file 5-6
ICF file 5-6
Work with Active Job (WRKACTJOB) command 6-3
Work with Configuration Status (WRKCFGSTS) command 4-1
Work with Device Table (WRKDEVTBL) command 3-5
Work with Job Queue (WRKJOBQ) command 6-3
Work with Program Table (WRKPGMTBL) command 3-7
Work with Table commands, non-ICF 3-5
Work with User Table (WRKUSRSTBL) command 3-6
write operation 5-5
writing finance application programs
considerations 7-6
ICF 5-1
non-ICF 6-1

X
X.21 circuit-switching network 2-1
X.25 communications line 3-1
X.25 packet-switching data network 2-1
Reader Comments—We’d Like to Hear from You!

AS/400 Advanced Series
Finance Communications
Programming
Version 4
Publication No. SC41-5449-00

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>
</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.