AS/400 Advanced Series

IBM

SNA Upline Facility Programming

Version 4

SC41-5446-00

AS/400 Advanced Series

SNA Upline Facility Programming

Version 4

SC41-5446-00

Take Note!

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

August Edition (August 1997)

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

Make sure that you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. If you live in the United States, Puerto Rico, or Guam, you can order publications through the IBM Software Manufacturing Solutions at 800+879-2755. Publications are not stocked at the address given below.

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

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

or you can fax your comments to:

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

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

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

Contents

Figures

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 publication is intended to help application programmers use the SNA Upline Function of the IBM OS/400 licensed program. This publication documents General-Use Programming Interface and Associated Guidance Information.

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

Trademarks

About SNA Upline Facility Programming, SC41-5446

This book provides the programming information you need to use Systems Network Architecture (SNA) upline facility (SNUF) with the IBM AS/400 system. This book also discusses the SNA 3270 program interface for SNUF that allows an AS/400 application to communicate with a host application by sending and receiving 3270 data streams. This book should be used with the book, *ICF Programming*. You should be familiar with the concepts explained in the *ICF Programming* book and apply those concepts to the information presented here for using SNUF.

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

Who Should Use This Book

This book is intended for application programmers for the AS/400 system, application programmers for the remote CICS/VS or IMS/VS system, and programmers for the remote system.

You should be able to program in the language you intend to use and be familiar with Systems Network Architecture (SNA) concepts and terminology. You should be familiar with the following information:

- The concepts of communications configuration described in the *Communications Configuration* book.
- The concepts of intersystem communications function (ICF) support described in the *ICF Programming* book.
- Operations and tasks described in *System Operation* book.
- AS/400 system programming (mainly work station) terminology.
- In some cases, terminology of the remote system.
- If you are using SNA 3270 program interface, you should be familiar with 3270 device emulation and data stream concepts.
- If you are using 3270 binary synchronous communications (BSC) device emulation, you should refer to the *IBM System/38 Data Communications Programmer's Guide*, SC21-7825, and the *IBM System/38 3270 Emulation Reference Manual and User's Guide*, SC21-7961.

Prerequisite and Related Information

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

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

The **SNA upline facility (SNUF)** provides distributed data processing to AS/400* users who want to communicate with a remote host system through Systems Network Architecture (SNA). The host system can be a System/370* computer, System/390* computer, 30xx, or 43xx processor using either **Customer Information Control System for Virtual Storage (CICS/VS)** or **Information Management System for Virtual Storage (IMS/VS)**. CICS/VS is a licensed program that operates on a host system, such as the System/370, which can be used in a communications network. IMS/VS is a general purpose system that enhances the capabilities of OS/VS for batch processing and telecommunication. It allows users to access a computer-maintained database through remote terminals.

SNUF handles the communications support needed to connect the AS/400 system to a host system. It allows you to write programs that can communicate with either CICS/VS or IMS/VS on a specific host system, without being concerned with the unique communications requirements of the host system. SNUF provides both an interactive and a batch communications interface between the AS/400 system and the host system.

SNUF Capabilities

SNUF has the following capabilities:

- AS/400 programs can start system tasks or user programs on host systems with CICS/VS or IMS/VS.
- CICS/VS and IMS/VS tasks on a host system can start programs on the AS/400 system.
- More than one program on an AS/400 system can communicate at the same time with CICS/VS or IMS/VS programs on a host system.
- SNUF can share a communications line with other SNA-based functions on the AS/400 system. See the *Communications Configuration* book for a description of the total number of lines available on the AS/400 system and for a list of the SNA-based functions available.
- Remote locations are defined for SNUF networks as part of the AS/400 configuration process.
- SNUF can run a single session per device. There can be more than one device per controller. A maximum of 255 SNUF devices can be attached to the same controller.
- Data lengths to be sent and received by your AS/400 system SNUF application program can be defined to a maximum of 32,767 characters.
- SNUF application programs can be written using any of the high-level languages (HLLs), Integrated Language Environment (ILE) C/400*, ILE COBOL/400*, ILE FORTRAN/400*, or ILE RPG/400* programming languages, together with the AS/400 data description specifications (DDS) keywords or the system-supplied formats.
- The SNUF 3270 support allows the AS/400 system to communicate with a System/370, System/390, 30xx, or 43xx host application by sending and receiving 3270 data streams.
- Communications between an AS/400 program and a host system program occurs in either half-duplex flip-flop or half-duplex contention modes.
- SNUF allows retail pass-through support when a host system is connected to the AS/400 system and the AS/400 system is connected to various retail controllers. (Retail communications supports the session between the retail controller and the AS/400 system. SNUF communications supports the session between the AS/400 system and the remote host system.) See the *Retail Communications Programming* book.

Communications Line Support

SNUF uses the following communications lines:

- Synchronous data link control (SDLC) lines
  - Point-to-point switched (manual answer, automatic answer, manual call, or automatic call)
  - Point-to-point nonswitched

  – Multipoint nonswitched

- X.25 lines
- Token-ring lines
- Ethernet lines
- Integrated services digital network (ISDN) data link control (IDLC) lines

  Note: When the host system physically resides on a token-ring local area network (LAN) it may communicate, using SNUF, to an AS/400 system physically residing on an Ethernet LAN if the two LANs are connected with an 8209 LAN bridge.

SNUF Communications Network

Figure 1-1 illustrates a SNUF network that communicates with several remote systems using different communications lines.

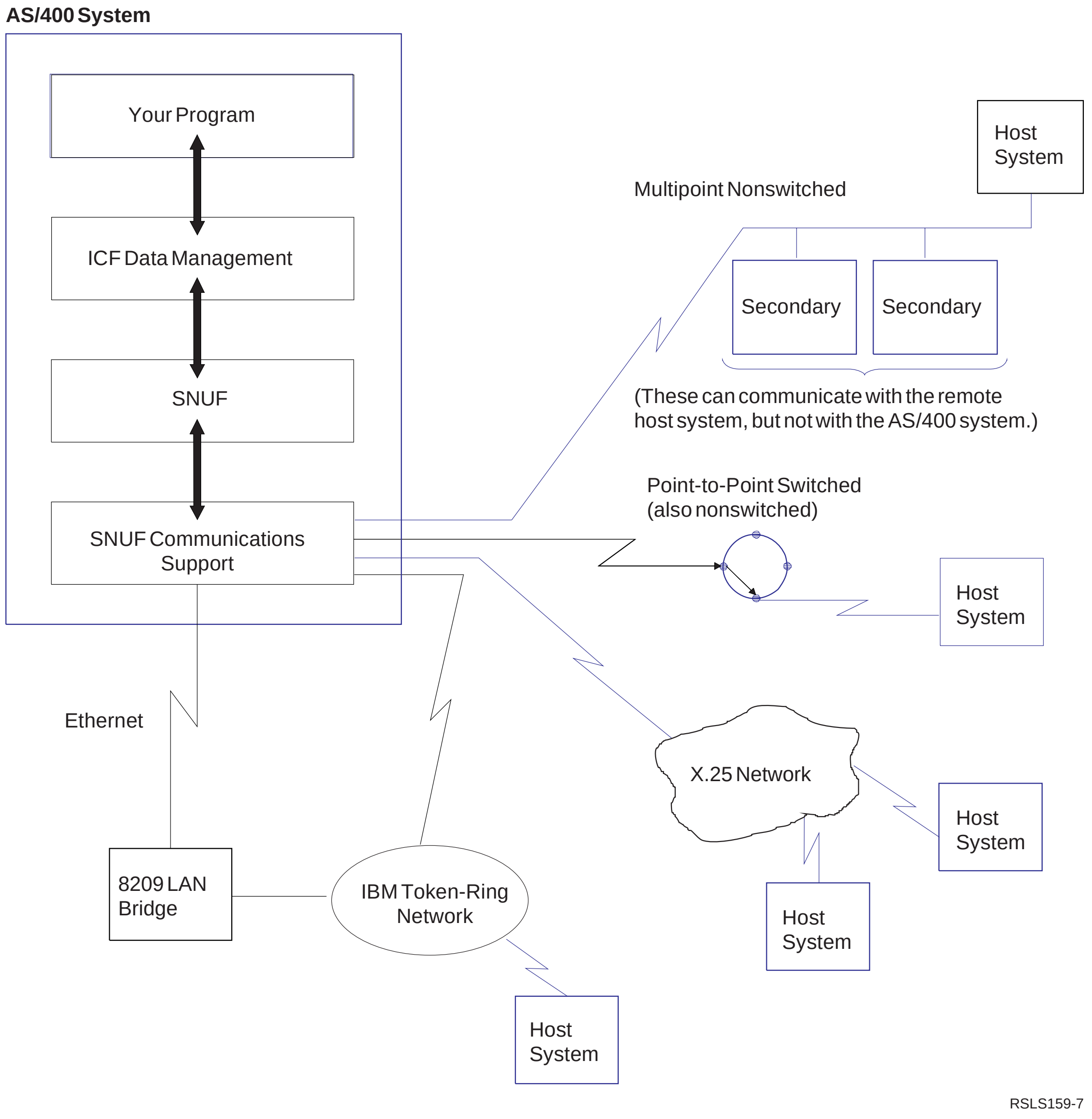


Figure 1-1. SNUF Communications Network

Chapter 2. Configuring SNUF Communications

This chapter discusses how to set up SNA upline facility (SNUF) by creating line, controller, and device descriptions.

Creating SNUF Descriptions

Before you can use SNUF for communications, you must create configuration descriptions for lines, controllers, and devices that will be used with SNUF.

AS/400 system support allows you to create more than one configuration description on the system. You use commands to configure SNUF support in two ways:

- Using the command prompt. Enter the command and press F4 (Prompt). A prompt menu appears for the command. Answer the prompts as described in the *Communications Configuration* book.
- Using direct entry. Enter the command and its parameters using the syntax described in the *CL Reference* book.

The following briefly introduces the commands you use to configure SNUF. Create these descriptions in the order presented. For a complete description of these and related commands, see the *Communications Configuration* book.

Creating a Line Description

The line description defines the communications line used to communicate with the remote system. Valid line types for SNUF communications are synchronous data link control (SDLC), X.25, token-ring network, Ethernet, and integrated services digital network (ISDN) data link control (IDLC).

The following commands allow you to create a line description for use with SNUF:

- Create Line Description (SDLC) (CRTLINSDLC)
- Create Line Description (X.25) (CRTLINX25)
- Create Line Description (Token-Ring) (CRTLINTRN)
- Create Line Description (Ethernet) (CRTLINETH)
- Create Line Description (IDLC) (CRTLINIDLC)

  If you are using an integrated services digital network (ISDN), a connection list and network interface description also need to be created and defined. The book, *ISDN Support* contains more information about configuring an ISDN network.

Creating a Controller Description

Create a controller description after you have created a line description. The controller description defines the characteristics of the remote system.

Use the Create Controller Description (SNA Host) (CRTCTLHOST) command to configure a controller for use with SNUF.

Creating a Device Description

Create a device description after you have created a controller description. The device description defines the characteristics of the device your application program is going to communicate with.

Use the Create Device Description (SNUF) (CRTDEVSNUF) command to create a device for a SNUF application.

Changing or Deleting a SNUF Description

Use the following commands to change a SNUF description:

- Change Line Description (SDLC) (CHGLINSDLC)
- Change Line Description (X.25) (CHGLINX25)
- Change Line Description (Token-Ring) (CHGLINTRN)
- Change Line Description (Ethernet) (CHGLINETH)
- Change Line Description (IDLC) (CHGLINIDLC)
- Change Controller Description (SNA Host) (CHGCTLHOST)

- Change Device Description (SNUF) (CHGDEVSNUF)

Use the following commands to delete a SNUF description:

- Delete Line Description (DLTLIND)
- Delete Controller Description (DLTCTLD)
- Delete Device Description (DLTDEVD)

Displaying a SNUF Description

Use the following commands to display a SNUF description:

- Display Line Description (DSPLIND)
- Display Controller Description (DSPCTLD)
- Display Device Description (DSPDEVD)

Chapter 3. Running SNUF Communications

This chapter contains the information you need to run your network, including information on the Vary Configuration (VRYCFG) command. See the book, *Communications Management* for additional information on running communications support.

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

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

The VRYCFG command with STATUS(*OFF) specified ends the link between two or more systems and releases the communications support. No further communication is possible between the systems until the specified configurations are varied on again.

For additional information on the Vary Configuration command, see the book, *Communications Management*.

Chapter 4. Writing SNUF Application Programs

This chapter describes how an application program uses the intersystem communications function (ICF) file and SNA upline facility (SNUF) communications support. The program can be coded using high-level languages (HLLs) that support an interface that allows the program to do the following functions:

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

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

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

Note: Before running application programs on your system, you must define the application environment. See the following publications for additional information:

- *Communications Configuration*
- *Communications Management*
- *ICF Programming*
- *Work Management*

Intersystem Communications Function (ICF) File

An **intersystem communications function (ICF)** file handles all communications between your program and the program on the remote system. The ICF file must be created before your application can use SNUF. For more information about the ICF file, see the book, *ICF Programming*.

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

The following commands are valid for the ICF file.

CRTICFF	Create ICF file and file-level attributes. This command allows you to create an ICF file.
CHGICFF	Change ICF file. This command allows you to change the file attributes of the ICF file.
OVRICFF	Override ICF file. This command allows you to temporarily change the file attributes of the ICF file at run time. These changes are only in effect for the duration of the job and do not affect other users of the file.
DLTF	Delete file. This command allows you to delete a file from the system.
DSPFD	Display file description. This command displays the file description of any file on the system. The information may be printed or displayed.
DSPFFD	Display field description. This command displays the description of the fields in any file on the system. This information may be printed or displayed.
ADDICFDEVE	Add ICF device entry. This command allows you to permanently add a program device entry that contains a program device name, remote location information, and session-level attributes.

CHGICFDEVE	Change ICF device entry. This command allows you to permanently change the program device attributes previously added with the ADDICFDEVE command.
OVRICFDEVE	Override ICF device entry. This command allows you to: • Temporarily add the program device entry, the remote location information, and the session-level attributes to the ICF file. • Override a program device entry with the specified remote location information and session-level attributes for an ICF file.
RMVICFDEVE	Remove ICF device entry. This command allows you to permanently remove the program device entry previously added with the ADDICFDEVE command or changed with the CHGICFDEVE command.

The commands CRTICFF, CHGICFF, and OVRICFF have a WAITFILE parameter. Use this parameter to specify the number of seconds that the program waits for the file resources to be allocated when the file is opened and a device is acquired. You must specify the number of seconds that the program waits for the file resources to be allocated. A value of 1 through 32 767 can be specified.

The commands ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE have specific parameters that are needed for SNUF communications. Figure 4-1 describes the SNUF parameters for the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands.

For a complete description of all the parameters for these commands, see the book, *ICF Programming*. See the *CL Reference* book for the syntax of the commands.

Figure 4-1. Command Parameters

Description	**ADDICFDEVE, OVRICFDEVE, or CHGICFDEVE Command Parameter**
File name (ADDICFDEVE and CHGICFDEVE only)	FILE
Program device name	PGMDEV
Remote location name	RMTLOCNAME
Communications type	CMNTYPE
Device description name	DEV
Input record selection method	FMTSLT
Host application identifier	APPID
Batch or interactive session flag	BATCH
Host subsystem type	HOST
End session with host	ENDSSNHOST
Input function management header handling method	HDRPROC
Special host application	SPCHOSTAPP
Initialize self-selection	INZSELF
Message protection flag	MSGPTC
Emulated device	EMLDEV
Maximum record length	RCDLEN
Maximum block length	BLKLEN
Override protection flag (OVRICFDEVE only)	SECURE

The following parameters have a special meaning for SNUF communications:

FILE

Specifies the name and library of the ICF file to which you are adding the program device entry. The FILE parameter is available only with the ADDICFDEVE and CHGICFDEVE commands.

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

***LIBL**: SNUF communications support uses the library list to locate the ICF file. This is the default value.

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

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

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

pgm-device-name: Type the name by which the user program will refer to this communications session.

RMTLOCNAME
Specifies the remote location name with which your program communicates. A remote location name must be specified on the ADDICFDEVE command or an OVRICFDEVE command. If a remote location name is not specified, return code 82EE is returned to your program when the program device is acquired.

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

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

Note: For additional information on how the remote location name is used for device selection, see the discussion of the DEV parameter later in this section.

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

***ALL**: This default value specifies that all parameters appear in the prompt.

***SNUF**: The prompt for all SNUF supported parameters.

Note: When you specify *REQUESTER for the remote location name (RMTLOCNAME), you are prompted for the attributes of the emulated device (EMLDEV), format select (FMTSLT), record length (RCDLEN), block length (BLKLEN), and secure (SECURE) parameters.

DEV
Specifies the communications device used in the remote location. This parameter must be specified for SNUF support.

***LOC**: The device will be determined by the system.

When *LOC is specified for the DEV parameter and the value for the RMTLOCNAME parameter is not *REQUESTER, then the device chosen is determined by the system according to the device selection criteria of SNUF.

SNUF acquires the first device with the specified remote location name that is varied on and not in use.

If no varied-on device is available, SNUF attempts to acquire a device in the vary-on pending state. If such a device is not available, SNUF chooses a device with a less favorable priority.

SNUF device selection is made, in the following priority, from devices with the specified remote location name that are not in use and are otherwise serviceable:

1. Varied on
2. Vary-on pending
3. In queried, suspend, or reset states
4. Recovery pending or inoperative pending state
5. Held immediate state
6. Held controlled state

device-name: Type the 1- to 10-character name of the device that is associated with the remote location.

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

***PGM**: The program determines what record formats are selected. This is the default value.

***RECID**: The RECID keywords specified in DDS for the file are used to specify record selection.

APPID
This parameter specifies the VTAM* identifier of the CICS/VS or IMS/VS host system.

***DEVD**: Specifies that the application identifier specified in the device description is sent with the logon message. This is the default value.

***USER**: Specifies that the application program receives the USSMSG message and sends a logon command to the host system. This is valid only when using the 3270 program interface.

application-ID: The specified application identifier is sent with the logon message.

BATCH
Specifies if batch processing is performed for the session with the CICS/VS or IMS/VS host system. See "Sending Records in Chains" on page 5-2 for a complete discussion of batch and interactive sessions.

***NO:** Specifies that the session will be running in interactive mode. This is the default value.

***YES:** Specifies that the session will be running in batch mode.

Note: If you specify RMTLOCNAME(*REQUESTER), this parameter is ignored. The program started by the remote system is always running in batch mode.

HOST
This parameter specifies the remote or host system with which this session is communicating.

***DEVD**: The host system specified in the device description is used. This is the default value.

***CICS**: The session communicates with CICS/VS.

***IMS:** The session communicates with IMS/VS.

***IMSRTR:** The session communicates with IMS/VS using the ready-to-receive option.

ENDSSNHOST
This parameter specifies the end of a session with the host system.

***RSHUTD**: Specifies the Request Shut Down command to the host system. Most host applications recognize this command. This is the default value.

***TERMSELF**: This value may be set to issue a TERM-SELF command to the host system. This value is used when the host application does not recognize the default value.

SPCHOSTAPP
Specifies whether SNUF should customize support for special host applications outside the CICS or IMS application layer.

***DEVD**: The value specified in the device description is used. This is the default value.

***NONE**: No customization for the special host application is needed.

***FLASH**: Customization for the Federal Reserve communication application, Federal Link Access for Secondary Half-sessions (FLASH), is required.

INZSELF
Specifies whether a formatted INIT-SELF is sent to the host system in place of an unformatted logon.

***NO**: Use the unformatted logon provided by SNUF. This is the default value.

***YES**: Use the formatted INIT-SELF provided by SNUF.

HDRPROC
This parameter specifies, for both CICS/VS and IMS/VS, if function management headers are passed to the application program.

***SYS**: SNUF removes function management headers before passing data to the program. This is the default value.

***USER**: Function management headers are passed to the program.

MSGPTC
This parameter specifies for both CICS/VS and IMS/VS whether message protection is used for this session.

***YES**: Message protection is used. SNUF saves messages until they are responded to, and uses these messages to determine if data must be sent again when a line, controller, or device failure occurs. This is the default value.

Transmission services profile 03 is not supported if message protection is requested.

Note: *YES is only valid when processing is not running in batch (BATCH(*NO) is specified).

***NO**: Message protection is not used.

EMLDEV
This parameter specifies whether or not the application is sending and receiving 3270 data streams.

***NONE**: This default value specifies that the program device entry is not used to send and receive 3270 data streams.

32xx: One of these values specifies that the device is used to send and receive 3270 data streams.

Note: If a value other than (*NONE) is specified, you are specifying that this device will send and receive 3270 data streams.

See “ICF File Considerations” on page D-1 for a complete description of this parameter and the SNA 3270 program interface for SNUF.

RCDLEN
This parameter specifies the maximum record length (in bytes) for the logical record of data being sent and received from your program. The specified length should not exceed the length of the input/output buffer, which is determined by the value of the MAXRCDLEN parameter on the CRTICFF command.

***DEVD**: The record length specified in the device description is used. This is the default value.

record-length: Type the maximum record length when using this device file. The value must be at least the size of the largest record sent and cannot exceed 32 767 bytes.

For SNA 3270 program interface, ensure that the specified length accommodates the larger 32 byte header, the largest display image possible with your application program, and possible field definitions that may follow the display or printer image.

Note: If a record is longer than the specified maximum record length, a run-time error occurs at the time the record is sent or received.

BLKLEN
This parameter specifies the maximum block length (in bytes) for data sent.

***DEVD**: The block length specified in the device description is used. This is the default value.

block-length: Specify the maximum block length of records sent or received when using this device file. The value must be greater than or equal to the value of RCDLEN, but cannot exceed 32 767 bytes.

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

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

***YES:** Specifies this program device is secure from previously called override commands.

Comparing Configuration and Program Device Entry Command Parameters

The parameter values from the configuration commands are used for any SNUF session, unless those values are changed by the program device entry commands.

Figure 4-2 on page 4-6 shows the relationship between the SNUF parameters for the program device entry commands (ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE) and the configuration commands. If there is no configuration parameter corresponding to the program device entry parameter, it is marked with a dash (–). Except where noted, you specify all configuration parameters when you create the device description (CRTDEVSNUF command).

The ADDICFDEVE and CHGICFDEVE program device entry commands cause permanent changes for any SNUF session using the specified program device. The OVRICFDEVE program device entry command causes job-level changes (as long as the OVRICFDEVE command remains in effect) for any SNUF session using the specified program device.

Figure 4-2. Comparing Configuration Command Parameters and Program Device Entry Command Parameters

Parameter Description	Configuration Command Parameter	Program Device Entry Command Parameter
File	–	FILE[1]
Program device name	–	PGMDEV
Device description	DEVD	–
Local location name	LOCADR	–
Remote location name	RMTLOCNAME	RMTLOCNAME[4,5]
Online at IPL	ONLINE	–
Communications type	–	CMNTYPE
Attached controller	CTL	–
Device	–	DEV[4]
Format select	–	FMTSLT
Program start request capable	PGMSTRRQS	–
Application identifier	APPID	APPID[3]
Batch activity	–	BATCH
Host type	HOST	HOST[3]
End session with host system	–	ENDSSNHOST
Special host application	–	SPCHOSTAPP
Initialize self	–	INZSELF
Header processing	–	HDRPROC
Message protection	–	MSGPTC
Emulation device	–	EMLDEV
Device type	–	device type
Data format	–	data format
Record length	RCDLEN	RCDLEN[3]
Block length	BLKLEN	BLKLEN[3]
Secure from override	–	SECURE[2]
Default program	DFTPGM	–
library	LIBRARY	–
Text description	TEXT	–
Authority	AUT	–

Notes:

1 This parameter is valid only on the ADDICFDEVE and CHGICFDEVE commands.

2 This parameter is valid only on the OVRICFDEVE command.

3 Specify (*DEVD) and the value is retrieved from the device description at run time. This is the default for the parameter. Specify (*USER) and the application program sends a logon message to the host system. The application program is also responsible for receiving and handling system services control point-logical unit (SSCP-LU) messages from the host system.

4 These parameters are used by the program device entry to associate a program device name with the device description you want to use. See the *ICF Programming* for more information on defining program device entries.

5 If you specify RMTLOCNAME(*REQUESTER) on the command, you are NOT prompted for these parameters: DEV, APPID, BATCH, HOST, HDRPROC, and MSGPTC. The program started by the host request will have acquired the device, so device selection using the RMTLOCNAME and DEV parameters does not occur. The value (*REQUESTER) is only valid on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands.

Communications Operations

This section gives a description of the operations you can code into a program using SNUF support to communicate with a host program.

For additional information on these operations and functions, see the book, *ICF Programming*. For coding examples, see the *DDS Reference* book.

Starting a Session

There are two ways to start a SNUF session:

- The AS/400 program can issue an open/acquire operation to establish a session between your program and a program at a remote location. See “Open or Acquire Operation” on page 4-7.
- The CICS/VS or IMS/VS program can send a program start request to the AS/400 system. See “Program Start Requests” on page 5-5.

Open or Acquire Operation

Your program uses the open or acquire operation to establish a session between your program and the host system.

You can start a session implicitly with the open operation when you specify the ACQPGMDEV parameter on the CRTICFF command, or you can start the session explicitly by using the acquire operation.

The open/acquire operation starts the following sequence:

- SNUF sends a sign-on to the host system using the application ID (APPID) parameter specified when you configured the program device entry (ADDICFDEVE) or changed the program device entry (OVRICFDEVE or CHGICFDEVE).
- SNUF examines the BIND command parameters from the host system.
- SNUF sends your program a normal return code when the host system is ready to begin the session.

For a switched connection, the open/acquire operation starts a similar sequence. Additional steps occur based on how the connection is configured. For more information, see the *Communications Configuration* book.

Sessions started by the acquire operation are started with the parameters specified in the ADDICFDEVE or OVRICFDEVE command. A parameter specified in the OVRICFDEVE command overrides any corresponding parameter specified in the ADDICFDEVE or CHGICFDEVE command.

Starting a Transaction

A **transaction** is a logical connection between two programs. Use the evoke function to start a transaction between your system and the host system. (A program start request from the remote host system is used to start a transaction between the host program and your program.)

Evoke Function

The evoke function starts a transaction and identifies the program on the CICS/VS or IMS/VS system that is to receive the data. You must acquire a program device to establish a session before you can issue an evoke function in your program. You can issue more than one evoke function in a SNUF session to send or receive multiple transactions to one or more remote programs. However, you cannot issue another evoke function in your program until the current transaction is ended by either issuing a detach function or receiving a detach indication. The evoke function is only supported when in half-duplex flip-flop mode.

The evoke function uses an evoke parameter list that identifies the remote program and, if security is being used by IMS/VS, the correct password. (For CICS/VS, security is handled with a special sign-on transaction.) This list can optionally contain user-supplied data for the remote program. If you use either the EVOKE DDS keyword or the system-supplied format ($$EVOK), possible parameters are:

- Remote program name
- User ID
- Library name
- User password
- User data or program parameters

Notes:

1. SNUF ignores the user ID, library name, and user password when communicating with CICS/VS, and ignores the user ID and library name when communicating with IMS/VS.
2. The optional data you can specify with each type of evoke function can be user data or program parameters.

When your program issues an evoke function, SNUF builds and sends a program start request to the host system. The host system receives this transaction start record (host terminology), which specifies an evoke parameter list containing the remote program name (as a transaction code), any password you specify, and any additional data you supplied. When the remote program starts, the host system passes the data you supplied to its program.

When CICS/VS receives the evoke request, it starts the program. When IMS/VS receives the evoke request, it schedules the specified program for running. IMS/VS specifies the transaction data when the program starts.

Sending Data

You can send data during a transaction using the write operation. Function management headers can also be sent with the write operation.

Write Operation

The write operation passes data records from your program to the remote program. Each write operation sends only one record from the SNUF application. To issue a write operation without sending data, specify a zero-length output record. (The zero-length output record tells SNUF there is no data associated with the write operation.) You can also use a write operation to send control information or commands (such as the IMS/SET command) to the host system. You can issue a write operation any time you have control of the session.

You can start a transaction with either a write operation or evoke function. If you start a transaction with a write operation, you are responsible for setting up the data to be sent.

The manner in which SNUF sends logical records depends on the BATCH parameter on the ADDICFDEVE or OVRICFDEVE command. If you specify BATCH(*NO), SNUF sends each logical record as a complete chain. SNUF automatically divides records greater than the size of the maximum request/response unit (RU) into elements of a chain. If you specify BATCH(*YES), SNUF divides logical records as required but does not chain them.

Variable length records can be sent by specifying the DDS keyword VARLEN for the record containing the variable length data. To set the length before a write operation, the length of the data contained in a field defined by this keyword in DDS can be accessed as required by your program. If your program combines the write operation with an input operation (for example, write-with-invite), SNUF sends a turnaround indication and performs the input operation. If your program issues a write-with-invite, it must issue a read or read-from-invited-program-devices operation to receive the data from SNUF. Use the timer function to limit the waiting time for the read-from-invited-program-devices operation.

If your program does not combine the write operation with an input operation (write, write with end-of-group, or write with detach), SNUF sends one data record to the remote system for each write operation.

If an error occurs while sending your data, SNUF notifies your program with a return code on the current write operation, and the data is not sent.

End-of-Group Function

Your program uses the end-of-group function to indicate that this is the last record in a chain of records. The end-of-group function does not indicate, however, that your program is ready to receive data. Use the DDS keyword allow write (ALWWRT) to indicate that your program has finished sending data and now wants to retrieve data from the host system.

Function-Management-Header Function

You can send a function management header (FM header) with a write operation. A function management header is valid only with the first record in a chain. SNUF checks whether the function management header is allowed, but does not check the function management header format or content.

The **function management header** is a special record or portion of a record that contains control information for the data that follows. The first byte of the record defines the length of the header. The length is specified in hexadecimal and includes the length byte. The header portion immediately follows the length byte.

When your program receives a function management header, the action SNUF takes depends on the header processing (HDRPROC) parameter of the ADDICFDEVE or OVRICFDEVE commands. If you specified HDRPROC(*SYS), SNUF removes the function management header and places any user data in the record area. If you specified HDRPROC(*USER), SNUF places the function management header in the record area. The return code indicates that a function management header was received. To receive any data that accompanied the FM header, issue a second input operation.

When a session started by a program start request receives a function management header, SNUF examines the FM header for hex 0542000001, which is the standard IMS/VS function management header. This function management header does not contain function management information and is not placed in the record area. If the function management header is not hex 0542000001, SNUF passes it to the application program as the first input. To get any additional data that accompanied the program start request, issue a second input operation.

Receiving Data

Two operations can be issued to receive data: read and read-from-invited-program-devices. Use the read operation to receive data from a specific program device, and use the read-from-invited-program-devices operation to read from any previously invited program device. Use the invite function to request data from a specific remote program.

Read Operation

Your program uses the read operation to obtain data from a specific program device. In a communications session, the operation causes SNUF to read data from the remote program with which your program is communicating. Your program does not receive control until the data is available.

The record that your program receives depends on the BATCH parameter of the ADDICFDEVE or OVRICFDEVE commands. If you specify BATCH(*NO), SNUF assembles each physical record into a logical record until it reaches the end-of-group indicator. Your program can perform record selection at the time a data record is received, based on the content of a portion of the record. Use the DDS keyword RECID to specify this selection value.

To check for an end-of-group indication sent by the host system, test a response indicator associated with the DDS keyword RCVENDGRP or test for one of the return codes listed under “Receive-End-of-Group” on page 4-14.

To check for a function management header on the first record of a chain of records, check for one of the return codes listed under “Receive-Function-Management Function” on page 4-14 or test a response indicator associated with the DDS keyword RCVFMH. If your program has received a function management header, your program must issue another read operation to obtain the data associated with that function management header.

Your program does not always receive data after an input request. In certain instances, only a return code is set to indicate a change in the operating state of a session. Test for a turnaround indication sent from the host system by testing a response indicator associated with the DDS keyword RCVTRNRND or by checking the return code.

Invite Function

Your program uses the invite function to request input data from a specific remote program. Your program receives control without waiting for the input. To obtain the data, your program must issue a read or read-from-invited-program-devices operation later in the transaction.

You can issue the invite function alone or in combination with another function.

Read-From-Invited-Program-Devices Operation

Your program can use the read-from-invited-program-devices operation to perform the following functions:

- Obtain data from any remote program that has responded to an invite function previously issued in your program. If data becomes available to your program from more than one remote program before the read-from-invited-program-devices operation is issued, your program receives the data that was first made available from a remote system.
- Verify that the time interval established by the timer function has run out, and if it has, ensure that your program is notified. For a

description of the timer function, see "Timer Function" on page 4-12.

All read-from-invited-program-devices operations should be issued to receive data after an invite function is issued by itself or in combination with another operation, or after a timer function is issued. The operation can receive any of the same return codes as the read operation.

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

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

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

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

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

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

Notifying the Remote Program of Problems

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

Fail Function

The fail function causes different indications to be sent to the host system, depending on the current state of your program.

- If your program is in a send state, SNUF sends a cancel indication to the remote host system. Any data not sent in the current transmission chain is discarded.
- If your program is in a receive state or if fail is the first function issued after receiving an end-of-group (chain) indication, SNUF sends a negative-response indication to the remote host system.

Cancel Function

Your program uses the cancel function to cancel the current chain of data it is sending to the remote program. The cancel function informs the remote program that it is abnormally ending the current data chain. The receiving program should disregard all records received since the last end-of-group (chain) indication.

The cancel function is valid only under the following conditions:

- While your program is in a send state.
- In a chain of records. You cannot cancel a chain after sending the end-of-group indication.
- In batch sessions. In batch sessions, a chain may contain several records.

Your program receives an error return code if it issues a cancel function in a session specified as BATCH(*NO) on the ADDICFDEVE or OVRICFDEVE commands. It also receives an error return code if it issues a cancel function between chains or in receive state.

The cancel function does not end the session. The host system can perform error recovery after receiving the cancel indication. To determine the recovery action the host system takes, issue an input operation after your program issues a cancel function.

The host system can also send a cancel indication to your program. To check for a cancel sent from the host system, check for return code 8330 or

8331, or test a response indicator associated with the DDS keyword RCVCANCEL.

Negative-Response Function

Your program uses the negative-response function when it detects an error with the data it received.

Issue the negative-response function when your program is in the receive state, the data received is in a chain, or the function is the first function after the end of a chain.

When your program sends a negative-response indication to the program that sent the data, it may include eight characters of sense code, which indicates the reason for the negative response.

The eight characters of sense code are coded as user data in your program output buffer. The first four characters in the buffer are the system sense code; the last four characters are the user sense code.

The system sense code must be one of the following: 10xx, 08xx, or 0000. SNUF checks the system sense code and rejects the operation if it is not one of the specified codes. If the program does not supply a system sense code, the system uses the default code of 0811 (break). The supported SNA sense codes are described in the *Systems Network Architecture Formats* book.

Your program can also receive a negative-response indication from the host system. To check for a negative response received from the host system, check for return code 8319 or test a response indicator associated with the DDS keyword RCVNEGRSP.

Using Additional Functions and Operations

Your program can use the get-attributes operation and the respond-to-confirm, request-to-write, cancel-invite, and timer functions with SNUF communications.

Respond-to-Confirm

Use the respond-to-confirm (RSPCONFIRM) keyword to send a positive response to a received definite response request. The respond-to-confirm function can be used only when a definite response request is outstanding. You can check the major and minor return codes or use the RCVCONFIRM indicator to determine when to issue a respond-to-confirm function. After sending the response, your program can continue processing as indicated by any other information received.

Request-to-Write Function

Your program uses the request-to-write function to indicate that it wants to send data to the remote program. When the remote program receives the request-to-write indication, it decides whether to stop sending data and when to stop.

After issuing this function, your program should continue to receive data until it receives a return code indicating the remote program is ready to begin receiving. (In some cases, the remote system may decide not to receive data and thus does not send a turnaround indication.) In response to the return code, begin sending data, perform other processing, or end your program.

Issue the request-to-write function only during a transaction and only when your program is in the receive state. If your program is neither receiving nor sending (that is, if it is between transactions), issuing the function has no effect and an 8327 code is returned to your program.

If the remote program sends a request-to-write indication, your program receives return code 0010 at the end of a write operation. If your program receives this return code, stop sending data and issue an input operation as soon as possible.

Cancel-Invite Function

Your program uses the cancel-invite function to cancel any invite function which has not received any input from any invited session. The cancel is handled by SNUF on the AS/400 system; no command or data is sent to the host system.

If data or a message is being received from the remote system when the cancel-invite function is initiated, the cancel-invite function fails and return code 0412 is received by your program. Your program must issue input operations to receive the data until it receives return code 0300 or 0308.

Timer Function

Your program can use the timer function before doing specified functions, such as a read-from-invited-program-devices operation. The timer function specifies an interval of time (in hours, minutes, and seconds) to wait before your program receives a return code of 0310 (timer run out).

Use the timer function to set the timer interval. The timer function is issued on an output operation to a record that has the record level keyword TIMER specified.

When a timer is set and your program requests data from a previously invited device, if data is available, your program receives the data along with a return code indicating a successful operation. If an error occurs, your program receives a return code describing the error. If the timer runs out before the data is received, return code 0310 is received by your program and the session remains invited.

Another way to specify the time interval is with the WAITRCD parameter on the CRTICFF, CHGICFF, OVRICFF commands. The WAITRCD parameter establishes the maximum time interval used for all read-from-invited-program-devices operations issued for the ICF file.

When the timer function is in effect, the value specified for the WAITRCD parameter is ignored.

Only one timer interval can be maintained for a program. If you set a new timer before an existing timer has run out, the new timer replaces the old one.

Note: For ILE RPG/400 programs, a timer function is not valid unless at least one session is attached to your program.

Get-Attributes Operation

Your program uses the get-attributes operation to determine the status of the current session. You can issue it at any time during the session. The operation gets the current status information about the session in which your program is communicating.

Ending a Transaction

A communications transaction can be ended by your program or by the program at the remote system.

Communications with the remote program ends when your program ends the transaction; however, the session may still exist:

- If your program acquired the session, either the AS/400 system or the host system can start another transaction in this session.
- If the host system acquired the session and your program ends the transaction, the remote system can issue another program start request to the AS/400 system. To begin another transaction from the AS/400 system, you must open or acquire a new session.

Detach Function

A transaction is ended with the detach function. The detach function informs the other program that your program is done sending data and has ended the transaction. The detach function is only supported in half-duplex flip-flop mode.

You can combine the detach function with either an evoke function or a write operation.

Ending a Session

A communications session is ended with either the release operation or the end-of-session function. You should primarily use the release operation because the release operation ends the session only if all processing is complete.

The end-of-session function always ends the session; therefore, it should be used only when you want to force the session to end.

Release Function

Your program uses the release function to end a session. This operation ends the session unless an error condition occurs (in which case, the release operation is not successful). To end the session unconditionally, use the end-of-session function.

If your program issues a release operation during an active transaction, it receives an error return code. The system performs the release operation only after all data for the transaction has been sent or received.

When your program issues a successful release operation, SNUF ends the session and frees the resources that were used by your program. The logical unit is made available to other programs in the system wanting to acquire the session or for another program start request from the host system.

End-of-Session Function

Your program uses the end-of-session function to end a session. The end-of-session function ends the session and always gives a normal completion return code. If your program uses the keyword during an active transaction, SNUF abnormally ends the transaction and the session, and possibly the remote program as well. However, your program still receives a normal completion code. You can use this function when an error occurs on a previous function and your program cannot easily recover.

When a program started by a program start request receives a detach return code, end the program or issue an end-of-session function. This leaves the session and remote program available for other transactions.

Note: A positive response is sent before the session is ended normally if the previous operation received an end-of-chain indication and a response is required. Negative responses are sent to the chains that are partially received by the user program, or if the session is ended by the system.

Using Response Indicators

Response indicators are defined to your program in the ICF file and are set on each input operation.

However, these indicators are optional, and major and minor return codes can also be used to indicate the status of input operations.

Receive-Cancel

Your program uses the receive-cancel response indicator to determine if the remote program canceled the current chain.

Receipt of a cancel request is also indicated by major return code 83 (session error) and minor return codes 30 (cancel with change-direction) or 31 (cancel without change-direction).

The cancel notification is always received without user data.

Receive-Confirm

Your program uses the receive-confirm response indicator to determine if the remote program sent an end-of-chain indication with the definite response request.

Receipt of a confirm request is also indicated by major return code 00 (user data received) and minor return code 03 (end of group received).

Receive-Detach

Your program uses the receive-detach response indicator to determine if the remote program has ended a transaction (the detach request has been received). The receive-detach function is only supported in half-duplex flip-flop mode.

The presence of the detach request is also indicated by major return codes 00 (user data received), 02 (user data received but program is being canceled), or 03 (no data received), and minor code 08 (detach received).

Receive-End-of-Group

The receive-end-of-group response indicator is used by your program to determine if your program has received the last record in a group (chain).

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

Receive-Function-Management Function

Your program uses the receive-function management header response indicator to determine that function management header data was received from the host program.

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

Receive-Negative-Response

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

Receipt of a negative-response function is also indicated by a return code of 8319.

Receive-Turnaround

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

Receipt of a turnaround function is also indicated by return codes 0000 (user data received), 0200 (user data received but program is being canceled), or 0300 (no data received).

Using the Input/Output Feedback Area

The results of input/output (I/O) operations are communicated to the program using the return codes, messages, and I/O feedback information. The area is changed for each I/O operation and consists of a common I/O area and a file-dependent I/O area.

Offset 48 in the file-dependant I/O feedback area applies to SNUF and indicates whether the remote program has requested permission to send data. For general information about the I/O feedback areas, see the book, *ICF Programming*.

Using Return Codes

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

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

For example, a major return code of 00 indicates that data was received. Along with this major code, you can receive from SNUF, for example, one of the following minor codes:

- 01: Indicates that your program should continue receiving data.
- 08: Indicates that the remote program has ended the transaction. Your program can do one of the following:
  - If your program acquired the session, issue another evoke function or end the session.
  - If the session was acquired by a program start from the remote system request, end the session and continue local processing or end the job.

Another example would be a major code of 83. In this case either the local system, remote system, or remote program has detected an error that may be recoverable. Different minor codes can be returned just as for the 00 major code. For example, if your program receives a minor return code of E8, your program has used a cancel-invite function in a session that was not invited. The cancel-invite function is only valid when it is used after a valid invite function. For this return code, your program is responsible for the necessary error recovery. The session and transaction are still active, and you can recover from this error by correcting the error in your program before trying to communicate with another program.

It is recommended that your program check the ICF return codes at the completion of every operation to ensure that the operation completed successfully or, if not, that the appropriate recovery action is taken.

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

Chapter 5. Considerations for SNUF

This chapter discusses programming considerations for application programs that provide communications between the AS/400 system and a host system. It examines programming topics for the AS/400 system programmers, and program start request formats for both the host and AS/400 system. See Appendix C for information needed by the host programmers to communicate through SNA upline facility (SNUF).

General Considerations

The following topics apply to both CICS/VS and IMS/VS host systems. They describe information needed by the AS/400 SNUF programmer while writing programs that communicate with either CICS/VS or IMS/VS. These topics also apply to both half-duplex flip-flop and half-duplex contention modes. This section is followed by a discussion of contention mode (see "Contention Mode Considerations" on page 5-4).

Half-Duplex Communications

Communication between an AS/400 program and a host system program occurs in either half-duplex flip-flop or contention modes, with one program sending at a time. When the sender wants to become the receiver, it sends a turnaround indication.

While sending, your program can cause a turnaround by issuing an input operation. SNUF interprets the input operation and sends the turnaround indication, as shown in Figure 5-1.

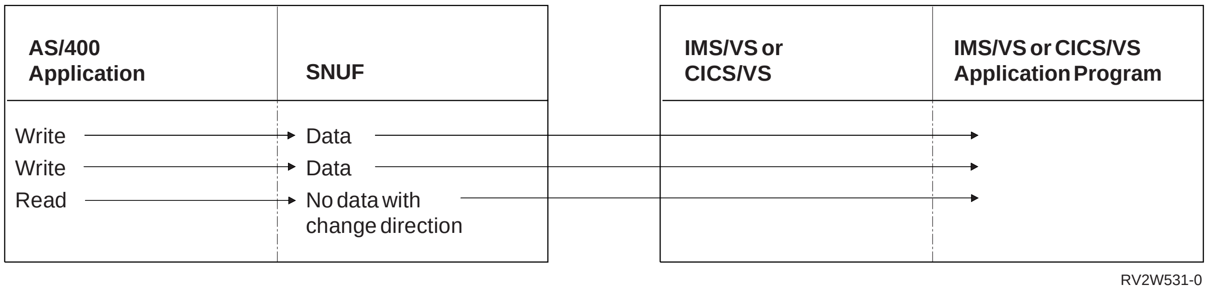


Figure 5-1. Sending Data in Half-Duplex Mode

To make more efficient use of the communications line, use write-with-invite to send a turnaround indication, as shown in Figure 5-2.

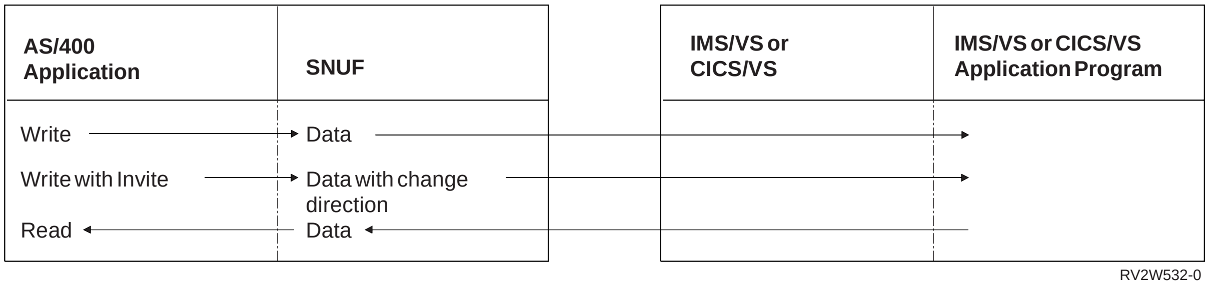


Figure 5-2. Sending a Change-Direction Indication in Half-Duplex Mode

If your program is receiving and must send data, use the request-to-write function. This causes SNUF to request that the current sender send a turnaround indication as soon as possible. If your program is sending data and receives a request-to-write return code, perform an input operation as soon as possible.

Sending Records in Chains

The request/response unit (RU) size parameter in the BIND command limits the size of a request unit that two logical units can send to each other. In order to send a request that contains more information than will fit into one RU, logical units divide the information into a series of separate requests. This series of related requests is called a chain. You determine the maximum length of an RU during host system generation.

How your program processes chains is determined by the type of session you specify: either an interactive or a batch session. You choose the type of session by specifying BATCH(*NO) or BATCH(*YES) on the ADDICFDEVE or OVRICFDEVE command.

The RU size parameter is ignored for a program started by a program start request. The program is treated as though BATCH(*YES) was specified.

Interactive Sessions: If you specify BATCH(*NO), each output operation is considered a logical record and is written as a separate chain. For input operations, SNUF assembles elements of a chain into one logical record until it reaches the end-of-group indication or the maximum record length. If SNUF reaches the maximum record length before it reaches the end-of-group indication, return code 81B9 is passed to your program and the session ends abnormally.

For output operations, SNUF sends each logical record as a chain. If the length of the logical record exceeds the size of the RU, a chain of request units is used to send the data. If the length of the logical record does not exceed the size of the RU, a single RU, with begin- and end-chain indicators, is sent.

The effect of chaining on the host IMS/VS or CICS/VS system must also be considered. If the host is not configured to assemble the chain of request/response units into a logical record, the host application programmer will be responsible for the task.

The following SNUF communications functions are not valid in interactive sessions:

- Cancel
- Cancel with invite

Figure 5-3 on page 5-3 shows how SNUF uses chains when you specify BATCH(*NO) and set the maximum request unit size to 256.

Batch Sessions: If you specify BATCH(*YES), SNUF does not attempt to distinguish logical records. For input operations, SNUF passes an RU for each read to the application program, which must determine the logical records. Therefore, while operating with BATCH(*YES), your program should check for an end-of-group return code.

Normally, when your program issues write operations and then performs an input operation or ends the transaction, it sends an end-of-group indication. In these cases, SNUF automatically ends the chain. When operating with BATCH(*YES), your program may want to send an end-of-group indication without sending a turnaround indication or without ending the transaction. For example, you may want your program to break data streams into smaller units that can be recovered. To accomplish this, issue a write-with-end-of-group function.

If your application uses the timer function, set the timer to the time:

- It will take the host to send, or
- For the AS/400 to receive all of the elements of the chain (first in chain, middle in chain, end of chain).

When the **entire** chain is received by SNUF, the chain can be retrieved by the application by using multiple READ operations.

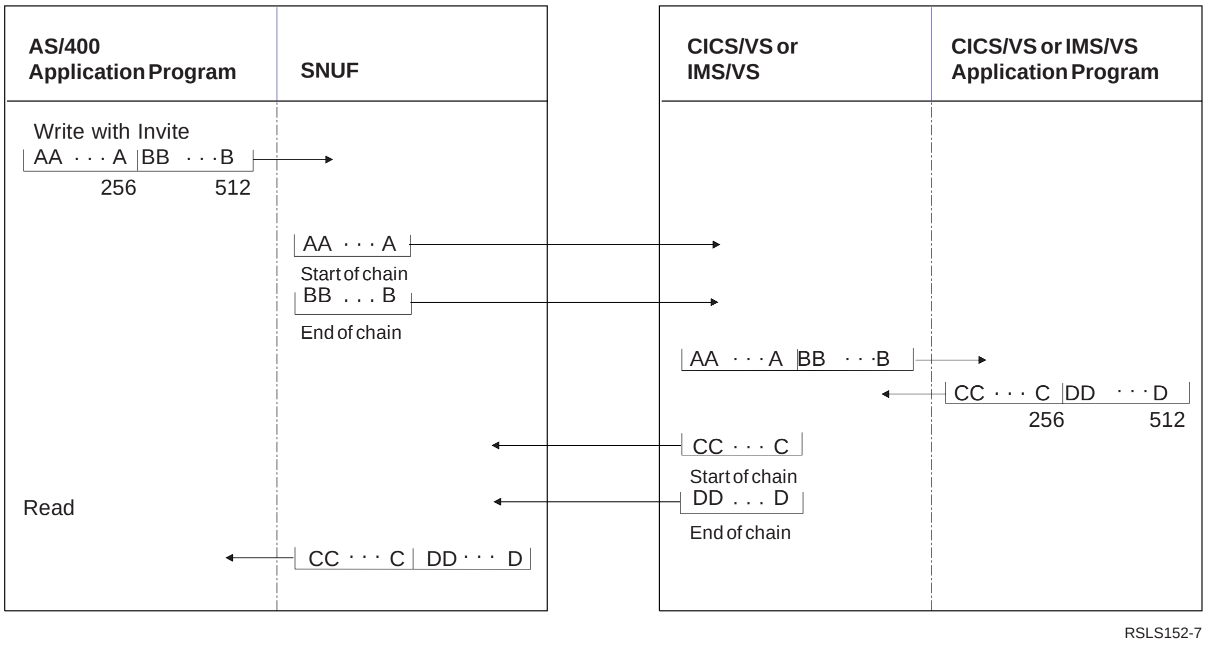


*Figure 5-3. Chaining in an Interactive Session – BATCH(*NO)*

Figure 5-4 shows how SNUF handles chaining when you specify BATCH(*YES).

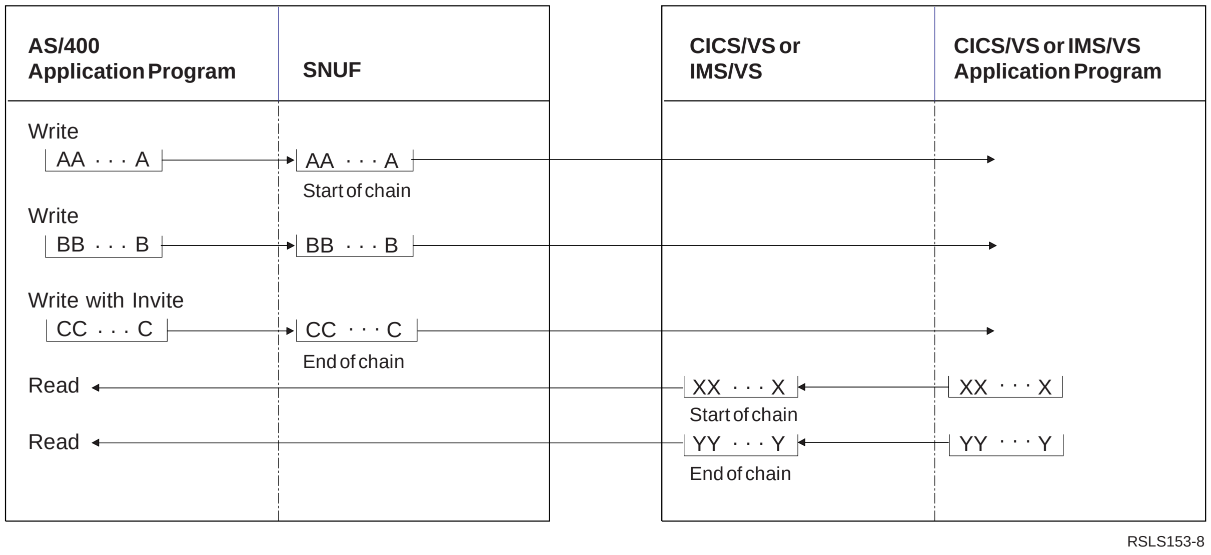


*Figure 5-4. Chaining in a Batch Session – BATCH(*YES)*

Receiving Messages from the Host System

SNUF can receive messages from both CICS/VS and IMS/VS. These messages inform SNUF and your program of key events occurring in the session. CICS/VS and IMS/VS send their messages in the following arrangement:

- For CICS/VS: DFHccnn text
- For IMS/VS: DFSccnn text

For CICS/VS, DFH identifies the message, and ccnn represents the message number as described in the *CICS/VS Messages and Codes*. For IMS/VS, DFS identifies the message, and ccnn represents the message number as described in the *IMS/VS Messages and Codes Reference Manual*. See the appropriate book for additional information about the received message.

When SNUF receives a host system message, it waits for your program to receive it on the next operation. If the next operation is an input operation, the message is returned to the input buffer of your program, and a return code is sent to your program to indicate that there is a message in the input buffer. If the next operation is not an input operation, the operation is rejected with a return code that indicates a message or data is waiting. In this case, your program must issue an input operation to get the message text. Until you receive the message, SNUF rejects any output operations.

CICS/VS and IMS/VS messages may be greater than the length of your program input buffer. If this occurs, SNUF truncates the message on the right and passes your program a return code indicating the truncation.

Session Recovery

Protected sessions can be successfully started again after communications line failures. You define a protected session by specifying MSGPTC(*YES) on the ADDICFDEVE or OVRICFDEVE commands. When SNUF starts the session again, it exchanges information with the host system about the last messages sent and received. SNUF uses this information to determine whether any data must be sent again. Therefore, to correctly start a protected session again, you must also define the host system session and transaction as protected. For CICS/VS host systems, specify the TYPE=OPTGRP parameter on the DFHPCT macro. For IMS/VS host systems, specify the INQUIRY parameter on the TRANSACT macro.

With a protected session, when a line error occurs SNUF tries to start the session again. If the session does not start again successfully, return code 8191 is passed to your program. If SNUF successfully establishes the session again, the session resumes at the point of failure.

Contention Mode Considerations

Communications between an AS/400 system program and a host system program can occur in half-duplex contention mode. The following is presented to help explain the differences between half-duplex flip-flop and contention modes.

- When communicating in flip-flop mode, the session is in contention state after the end-bracket. When communicating in contention mode, the session is in contention state after the end-chain.
- When in contention, SNUF is the contention winner.
- Since begin-bracket or end-bracket protocol is not used when communicating in contention mode, EVOKE, DETACH or RCVDETACH functions are not supported.
- The first request (RU) after the BIND command is treated as a program start request. The program start request formats described in Figure 5-5 on page 5-6 and Figure 5-6 on page 5-6 are also applicable in half-duplex contention mode. The session should be ended and then restarted to run another program on the AS/400 system.

Performance Considerations

The following suggestions can help you get better performance in sessions using SNUF:

- Combine input operations with output operations. For example, use evoke-with-invite.
- Specify a nonprotected session with MSGPTC(*NO) on the ADDICFDEVE or OVRICFDEVE commands. Nonprotected sessions require less line activity than protected sessions; however, no session recovery is provided by SNUF.
- Specify an RU size large enough to contain the largest record to be sent or received. Specify the RU size as large as the value specified for the maximum user record length. Larger RU sizes may improve performance. For more information on blocking, see the book, *Communications Management*.
- Specify a proper pacing count. If pacing is needed, seven is the best pacing count, as

values above this might not improve performance. For more information on pacing counts see the book, *Communications Management*.

- The host system configuration parameters MAXDATA, MAXOUT, and PACING, the BFRS Group Macro parameter, and the PASSLIM Build Macro parameter can affect communications performance. For more information, see “Performance Considerations” on page C-5.

Program Start Requests

For a program on a CICS/VS or IMS/VS remote system to start a program on the AS/400 system, the remote program must send a program start request to the AS/400 system after you have started communications between the two systems. The program started by the program start request must run on a device you specify at configuration time as being program start request capable.

When SNUF receives a program start request, it determines if the job should be run in the System/36 environment. If a System/36 procedure cannot be found SNUF starts the job in the OS/400 environment. For compatibility, the formats *TXTC, *TXTX, *EXEC, and *EXEX can be used to start a job in either the System/36 or the OS/400 environment. The following sections only consider the OS/400 environment. For running jobs in the System/36 environment, see the *System/36 Environment Programming*.

Communicating programs started by the host system (by the *EXEC, *EXEX, *TXTC, or *TXTX program start request) are treated as if BATCH (*YES) is specified. This means that each input request from the AS/400 program is satisfied with one element of a chain instead of requiring the entire chain. The end-of-chain return code is set when the last element of the chain is received, if it is not overridden by the end-of-transaction or change-of-direction return code.

Note: For general information on writing programs to be started by a program start request, see the book, *ICF Programming*.

Formats of the Program Start Request: CICS/VS and IMS/VS on a remote system can send four different program start request formats:

Format	**Description**
*TXTC	This format starts a session in which CICS/VS or IMS/VS can send more than one record to the same program on the AS/400 system before the session is ended.
*TXTX	This format starts a session in which the request statement is the only source of data for the program or it is the only source of parameters for the program.
*EXEC	This format functions the same as the *TXTC format but is used for System/36 compatibility only.
*EXEX	This format functions the same as the *TXTX format but is used for System/36 compatibility only.

The *TXTC and *TXTX formats allow you to use up to a 10-character program name and library name. The *EXEC and *EXEX formats allow only an 8-character program name and library name. The *EXEC and *EXEX formats are the same formats used on the System/36 and are included for use by CICS/VS and IMS/VS programs which formerly communicated with System/36 programs using SNUF. Use the *EXEC and *EXEX formats for applications requiring compatibility with System/36, and use the *TXTC and *TXTX formats for all other applications.

The program start request identifies which program is to be started. The request can include up to 119 bytes of parameters if the format is *EXEX or *EXEC, and 218 bytes of parameters if the format is *TXTX or *TXTC to be passed to a program. In the System/36 environment, these parameters can be treated as data and received by the application program using the first read.

A program start request must be on the first request unit (RU). Once the AS/400 program receives all the data, issue an end-of-session function. Otherwise, the session does not end until the AS/400 program ends.

A session started by IMS/VS with a program start request can pass data or parameters with the request but it cannot receive data from the AS/400 system. Similarly, a program on the AS/400 system can receive data from IMS/VS in a remotely started session but cannot send data.

Keep the session active until the AS/400 program receives the data and a detach return code. Then end the session or end the program. To have the AS/400 program communicate further with IMS/VS, include an OVRICFDEVE command and issue an acquire operation to acquire a new session with IMS/VS.

The begin-bracket (in half-duplex flip-flop mode) and first-of-chain indicators must accompany the program start request. If the program start request is a detach (*TXTX, *EXEX) request, end-bracket (in half-duplex flip-flop mode) and end-of-chain indicators must also accompany the request.

The host system must send a BIND command to logical units reserved for program start requests on the AS/400 system. Use the VTAM VARY command with the LOGON option, the LOGAPPL parameter in the VTAM definition, or the appropriate host system procedure (CICS/VS ACQ master terminal command or the IMS/VS /OPNDST command). After a detach return code, the AS/400 program should issue the end-of-session function so other program start requests can be handled.

A program started by a program start request is always run in batch mode. It is treated as if BATCH(*YES) is specified on the ADDICFDEVE command.

Syntax of the Program Start Request Statement: The type of program start request (*EXEC, *EXEX, *TXTX, or *TXTC) must begin in position 1 of the program start request statement. If the type begins in any other position, SNUF starts the default program instead of the program named in the statement.

The syntax of the program start request statement is shown in the following diagram:

```
*TXxx or *EXxx program name
   ┌            ┐  ┌                 ┐
   │ parameters │  │ user identifier │
   └            ┘  └                 ┘

   ┌              ┐  ┌               ┐
   │ library name │  │ user password │
   └              ┘  └               ┘
```

Figure 5-5 describes each parameter for a program start request type of *TXTX or *TXTC.

*Figure 5-5. Parameters for the Program Start Request (*TXTX, *TXTC)*

Coding Positions	Field	Description
1 through 6	*TXTX, *TXTC	Type of program start request being used to start a program on the AS/400 system. Position 6 must be a blank.
7 through xx	Program name	The name of the program to be started on the AS/400 system. The name must be 1 to 10 characters long. One or more blanks must follow the name.
xx through 226	Parameter	Parameter for the program started. This field begins with the first nonblank character following the program name.
227 through 236	User ID	The user ID (name) of the AS/400 user whose program is being started. If security is active on the AS/400 system, this ID must be defined on the system.
237 through 246	Library name	The name of the AS/400 library that contains the program to be started.
247 through 256	User password	The password of the AS/400 user whose program is being started.

Figure 5-6 on page 5-6 describes each parameter for a program start request type of *EXEX or *EXEC.

*Figure 5-6. Parameters for the Program Start Request (*EXEX, *EXEC)*

Coding Positions	Field	Description
1 through 6	*EXEX, *EXEC	Type of program start request being used to start a program on the AS/400 system. Position 6 must be a blank.
7 through xx	Program name	The name of the program to be started on the AS/400 system. The name must be 1 to 8 characters long. One or more blanks must follow the name.
xx through 127	Parameter	Parameter for the program started. This field begins with the first nonblank character following the program name.
128 through 135	User ID	The user ID (name) of the AS/400 user whose program is being started. If security is active on the AS/400 system, this ID must be defined on the system.
136 through 143	Library name	The name of the AS/400 library that contains the program to be started.
144 through 147	User pass-word	The password of the AS/400 user whose program is being started. The password must contain 4 characters. If security is active on the AS/400 system, this password must be defined on that system and must be the correct password for the user ID specified.

Note: The user ID, library name, and user password fields are positional and must be padded on the right with blanks if another field follows. If security is not used on the AS/400 system, the user ID and password are not required.

The program start request statement can contain parameters following the program name. Any parameter that follows the program name through position 127 if the format is *EXEX or *EXEC, or position 226 if the format is *TXTX or *TXTC, is used by the program started on the AS/400 system. If security is used, the remote system uses the positional parameters specified in positions 128 through 147 for the formats *EXEX or *EXEC or positions 227 through 256 for formats *TXTX or *TXTC to pass security information to the AS/400 system.

At least one blank must separate the program name that begins in position 7 from the data or parameters. If the AS/400 system uses security, send the program start request as a 147-byte record if the format is *EXEX or *EXEC, and as a 256-byte record if the format is *TXTX or *TXTC. Any unused positions should contain blanks.

If the system does not use security and does not specify a library name, you do not need the three parameters in positions 128 through 147 for the formats *EXEX or *EXEC or positions 227 through 256 for formats *TXTX or *TXTC. In this case, the length of the program start request depends only on the amount of data and the number of parameters to be passed to the AS/400 program.

If a program was not started successfully, a negative response with sense data is sent to the remote system. The sense data contains the reason code of the failure.

Sample of a Program Start Request:
Figure 5-7 shows a sample of the record format of a CICS/VS or IMS/VS program start request.

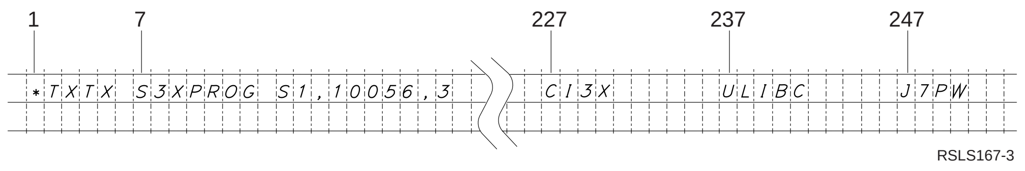


Figure 5-7. Example Program Start Request Record Format

In Figure 5-7, the program start request starts the AS/400 program named S3XPROG and sends it three positional parameter values. The identifier of the AS/400 user whose program is being started is CI3X. The user's password is J7PW. The S3XPROG program is located in the AS/400 library named ULIBC.

Prestart Jobs Considerations

To minimize the amount of time required to carry out a program start request, you can use prestart jobs to start a job on your system before the remote system sends a program start request.

To use prestart jobs, you need to define both communications and prestart job entries in the subsystem description, and make certain programming changes to the prestart job program with which the host program communicates. For information on using prestarted jobs, see the book, *ICF Programming*.

Programming for CICS/VS Systems

The following topics describe information needed by the AS/400 SNUF programmer while writing programs that communicate with CICS/VS. This information describes half-duplex flip-flop communications.

Evoke Considerations for a CICS/VS System

When it communicates with CICS/VS, SNUF ignores the user ID, library name, and user password parameters in the evoke function and sends the remote program name and the user data or program parameter. The first parameter is both the name of the CICS/VS program to be evoked and the CICS/VS transaction code. The name of the CICS/VS program is limited to four characters. SNUF does not send the user password, library name, or user ID parameters specified in the program evoke function.

If the CICS/VS host system is using security, the first evoke function your program issues must start a sign-on (CSSN) transaction on the host system. You must include the required security information with the evoke function. You also must specify the transaction code CSSN in the program name parameter and include the keyword parameters PS (user password) and NAME (user ID) as the program parameter. For additional information about the CSSN and sign-off (CSSF) transactions, see "SNUF Transaction Codes" and "Security Considerations."

SNUF Transaction Codes

There are several transactions that an AS/400 program can send to a CICS/VS system to start a particular remote program. An AS/400 program can use an evoke function to specify the CICS/VS service routine or application program to be started. The transaction codes in Figure 5-8 can be specified in the procedure name parameter of an evoke function.

Figure 5-8. SNUF Transaction Codes

Transaction Code	Purpose of Transaction
CSSN	Starts a CICS/VS program that controls the security of a CICS/VS host system and allows the user of the AS/400 program to sign on to the host system.
CSSF	Starts a CICS/VS program that signs off a user who has finished communicating with the host system.

Security Considerations

CICS/VS provides security for the work station operator rather than providing security for the device used by the operator. The CICS/VS security support is handled by two CICS/VS transactions: CSSN (sign-on) and CSSF (sign-off). The CSSN and CSSF transactions are used only to start and end a session in which security protects the transactions that occur in the session. SNUF can issue an evoke function to start the security transactions and the user transactions to be protected. User transactions must be started (with an evoke function) between the use of CSSN and CSSF transactions.

For additional security, supply the password to the AS/400 system from an external source, such as a work station operator.

CSSN Transaction: The CSSN (sign-on) transaction signs a user on to the CICS/VS host system. To evoke the CSSN transaction on a CICS/VS system with active security, the evoke function must include two security parameters. The parameters are user password ("PS"=) and user ID ("NAME"=), which are specified in keyword form and separated by a comma. The password can be 1 to 4 characters and the user ID can be 1 to 20 characters.

The evoke function for the CSSN transaction always results in a reply from CICS/VS. To receive the reply message, your program must issue an input operation after it issues the evoke function.

If an AS/400 program issues a successful CSSN evoke function, it must issue a CSSF evoke function before it issues a release operation in that session. If the program issues a release operation without a CSSF evoke, CICS/VS signs the user off the host system.

CSSF Transaction: The CSSF (sign-off) transaction ends communications between your program and CICS/VS. The CSSF transaction also removes the previously specified password and name from the CICS/VS sign-on table.

After the evoke function for the CSSF transaction has been completed, a CICS/VS message is available to your program. Issue an input operation to receive the message.

After a successful CSSF evoke function, the program can issue a CSSN evoke function to the same session or to another session. The function can use a different password and name each time.

For more information about the CSSN and CSSF CICS/VS transactions, see the *CICS/VS Supplied Transactions* book.

Programming for IMS/VS Systems

The following IMS topics describe information needed by the AS/400 SNUF programmer while writing programs that communicate with IMS/VS. This information describes half-duplex flip-flop communications.

Evoke Considerations for an IMS/VS System

When communicating with IMS/VS, SNUF does not send the user identifier and library name parameters because they are not used by IMS/VS. SNUF sends the remote program name, the user password (if it is specified), and the user data or program parameters. If IMS/VS uses security, you must specify the user password in the parameter list. To send the user password to IMS/VS, you also must specify *IMS or *IMSRTR on the HOST parameter of the ADDICFDEVE or OVRICFDEVE command.

Sending IMS/VS Commands

The AS/400 program can send IMS/VS commands by using the write operation and placing the command at the beginning of the logical record buffer. The system can only send commands when the session is between transactions. The program should not send commands that alter the status of the logical unit (such as the /ASSIGN command) because the results cannot be predicted.

IMS/VS Message Headers

If you specify HDRPROC(*USER) on the ADDICFDEVE or OVRICFDEVE commands, SNUF passes IMS/VS message headers to the program in its input buffer. The program also can send message headers by using the function management header function (see “Function-Management-Header Function” on page 4-8). Message headers can be used to pass message descriptors, component identification and control information. Additional information on message headers can be found in the *IMS/VS Advanced Function for Communications* book.

Security Considerations

If IMS/VS requires password security from the AS/400 program, the AS/400 program can supply the password in the evoke parameter list. SNUF sends the password in the correct position on behalf of the user. For additional security, supply the password to the AS/400 program from an external source, such as a work station operator.

Handling Errors

When IMS/VS detects an error on a received message, it returns an exception response with sense data. SNUF notifies the AS/400 program that an error has occurred and that sense data is available. To receive the sense data and the status of the session, issue an input operation.

The system sense bytes are either hex 0800 or hex 0826. The user sense bytes contain the IMS/VS error message number in hexadecimal form. For example, if the AS/400 program evokes an invalid transaction identifier, IMS/VS returns sense bytes of hex 08000040. This is converted and placed in the program buffer as the characters DFS0064.

Sending Transactions without Waiting for Output

An AS/400 program can use the evoke-with-detach function to send a complete transaction to IMS/VS without waiting for output from the IMS/VS program. This capability allows the AS/400 program to send several transactions before receiving a reply from IMS/VS.

An order entry application uses this type of processing, as shown in Figure 5-9. The evoke-with-detach function also allows the program to evoke a transaction in which the program reply is sent to another session, such as a program start session.

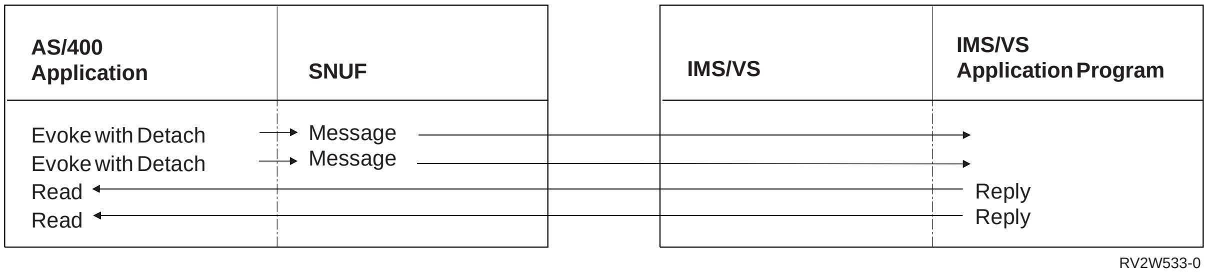


Figure 5-9. Sending Transactions without Waiting for IMS Output

The processing may differ slightly if you specified HOST(*IMSRTR) on the ADDICFDEVE or OVRICFDEVE command, as shown in Figure 5-10 on page 5-10.

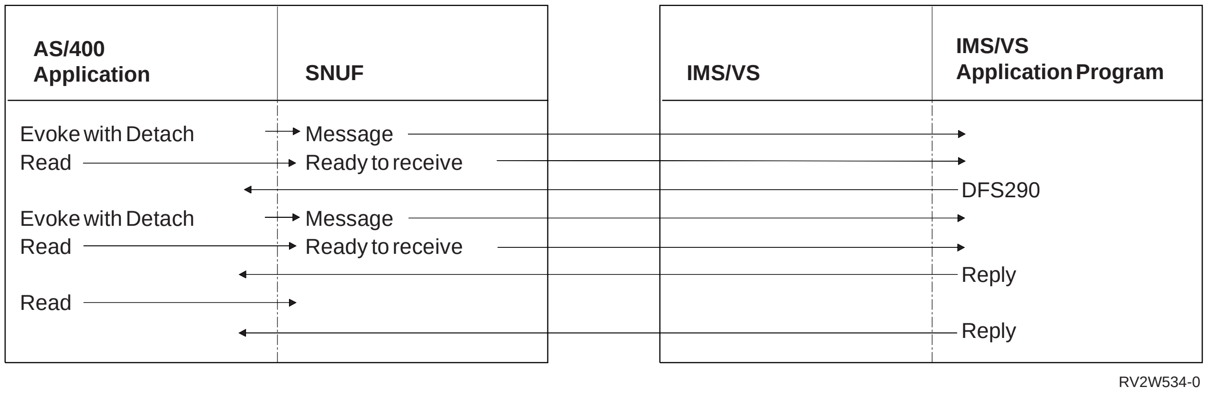


Figure 5-10. Sending Transactions without Waiting for IMSRTR Output

Requesting Messages with the Ready-to-Receive Command

The SNA ready-to-receive (RTR) command can be used to request any messages waiting on the IMS/VS output queue for this session. Before you send this command, you must specify HOST(*IMSRTR) on the ADDICFDEVE or OVRICFDEVE command. SNUF issues the RTR command when the AS/400 program issues an input operation (read) *between* transactions.

If a message exists on the IMS/VS output queue when SNUF sends the RTR command, IMS/VS sends the message. If no messages exist on the output queue, IMS/VS sends system message DFS290, which indicates the queue is empty. SNUF passes the DFS290 message to the AS/400 program with return code 0028, indicating a system message with a detach indication was received. If you expect more data, continue with other processing and try the input operation again later.

If you specified HOST(*IMS) on the ADDICFDEVE or OVRICFDEVE command, the AS/400 program must wait until output is available before it can complete the input operation.

Operating in Terminal Response Mode

Terminal response mode is an operation method defined for transactions and terminals attached to IMS/VS. When an IMS/VS logical terminal (LTERM) operates in terminal response mode, each transaction evoked must have a reply before the next transaction can be evoked.

Figure 5-11 on page 5-12 shows how an inquiry program on the AS/400 system starts an inquiry program on the IMS/VS system. The figure shows both terminal response mode and nonterminal response mode.

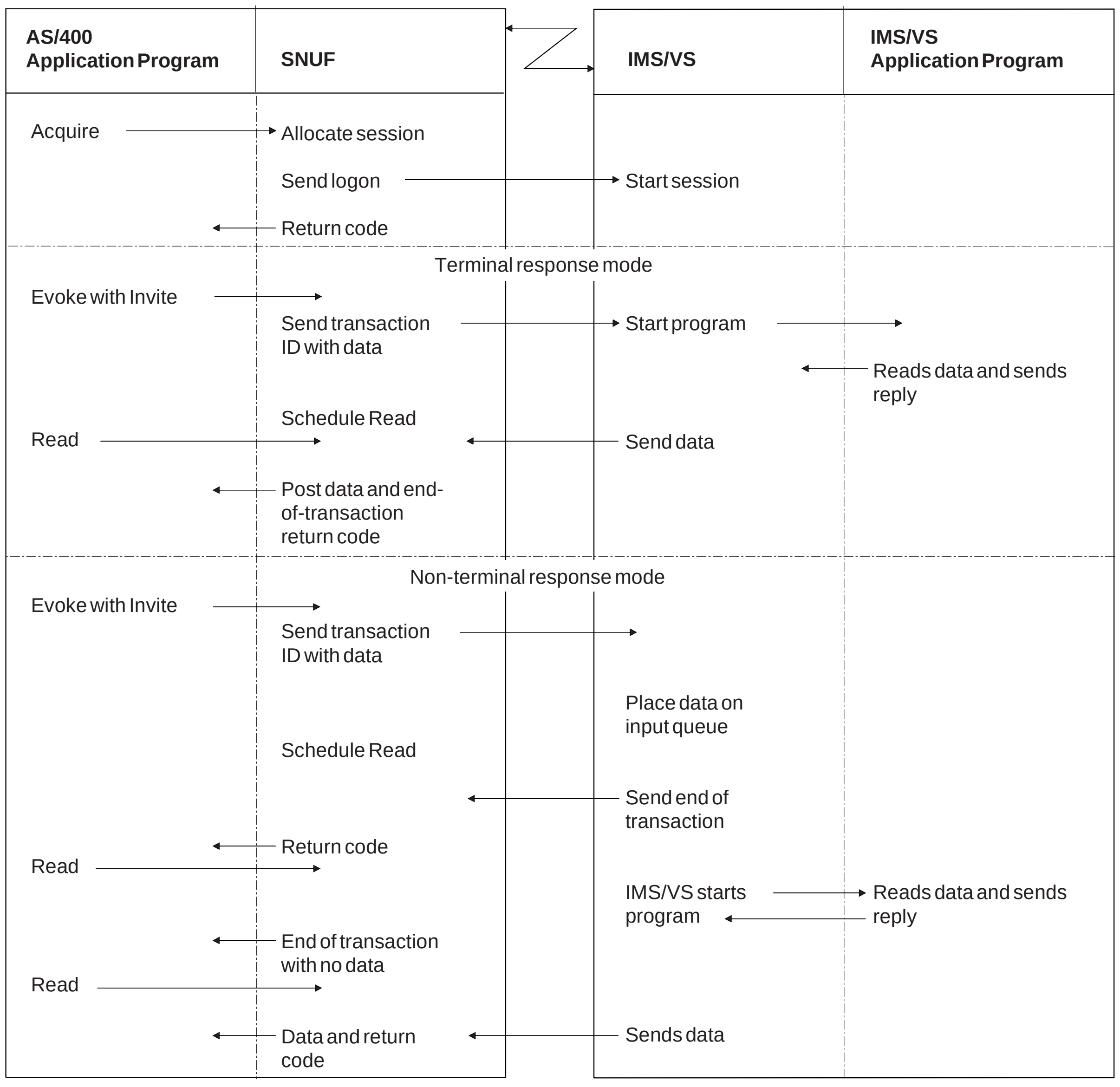

AS/400 Application Program
SNUF
IMS/VS
IMS/VS Application Program
Acquire
Allocate session
Send logon
Start session
Return code
Terminal response mode
Evoke with Invite
Send transaction ID with data
Start program
Reads data and sends reply
Schedule Read
Read
Send data
Post data and end-of-transaction return code
Non-terminal response mode
Evoke with Invite
Send transaction ID with data
Place data on input queue
Schedule Read
Send end of transaction
Return code
Read
IMS/VS starts program
Reads data and sends reply
End of transaction with no data
Read
Data and return code
Sends data
RSLS163-5

Figure 5-11. Operating in Terminal and Non-terminal Response Mode

You describe the terminal response mode with one of the following attributes:

Negated	Terminal response mode is not used for any transaction.
Forced	Terminal response mode is used for every transaction.
Transaction	Terminal response mode is defined separately for each transaction.

Operating in negated terminal response mode allows an AS/400 program to evoke several transactions before receiving a reply from any one of them. The program does, however, require more processing because it must correlate replies from IMS/VS to the transactions that created them.

For example, if the program issues an evoke-with-invite followed by a read operation, the return code indicates no data and end of transaction, or it indicates data (a reply) and end of transaction. In either case, another transaction can be evoked, but the session should not end until all replies have been received. After all transactions have been evoked, issue input operations until the program receives all replies.

If the program releases the session or ends before it receives all replies, data remains on the IMS/VS output queue. Data that cannot be recovered that is left on the IMS/VS output queue may be lost when the program ends a session. Data put on the queue after the program releases the session becomes the first input record received when another AS/400 application program acquires the session.

Operating in forced terminal response mode might require less processing than operating in negated or transaction mode. This is because IMS/VS does not allow input in a session until the system sends the reply to the previously evoked transaction. A program that evokes a transaction in a session using terminal response mode cannot use the same session until it receives the reply. If an error prevents the system from sending a reply, the session waits for data until the session abnormally ends. Some conditions that might prevent the system from sending a reply message are:

- LTERM has stopped.
- IMS/VS is unable to schedule the message processing program.
- A message processing program logic error prevents a message from being sent.

Using Message Format Services to Improve Performance

Message format services can give IMS/VS programs independence from terminal requirements and can improve online performance. If LTERMS are to use message format services, the service must be defined during IMS/VS creation. Message format services processing begins when one of the following occurs:

- The AS/400 program requests message format services by sending a function management header which contains a message identifier (midname).
- The AS/400 program sends *// midname* before sending a message.

Output messages from IMS/VS that are processed by message format services are sent with a function management header. To have the AS/400 program process these headers, specify HDPROC(*YES) on the ADDICFDEVE or OVRICFDEVE command.

For a complete description of message format services, see the *IMS/VS Message Format Service User's Guide*.

BIND Considerations for AS/400 Applications Using SNUF

Refer to SNA Distribution Services, Appendix D, for information about using the VM/MVS Bridge.

Appendix A. Language Operations, DDS Keywords, and System-Supplied Formats

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

- All valid communications operations supported by the intersystem communications function (ICF) file.
- All valid communications operations supported and the associated high-level language operations.
- Data description specifications (DDS) processing keywords.
- System-supplied formats.

Intersystem Communications Function Operations

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

Figure A-1. SNUF Supported ICF Operations

Operation	Functional Description
Open	Opens the ICF File.
Acquire	Establishes a session between the application and the remote location.
Get-attributes	Determines the status of the session.
Read	Receives data from the remote system.
Read-from-invited-program-devices	Receives data from an invited program device.
Write	Sends data records from the issuing program to the other program in the transaction.
Write/Read	Allows a write operation followed by a read operation. Valid for ILE C/400 and ILE RPG/400.
Release	Attempts to end a session.
Close	Closes the ICF File.

Language Operations Supported

Use high-level language operations and ICF to communicate with a program at a remote system. (See the specific high-level language book for operations other than ICF.) Figure A-2 presents the ICF file operations used with SNUF communications and the equivalent high-level language statement.

Figure A-2 (Page 1 of 2). High-Level Language I/O Operations

ICF Operation	ILE RPG/400 Operation Code	ILE COBOL/400 Procedure Statement	ILE C/400 Function[2]	ILE FORTRAN/400 Statement
Open	OPEN	OPEN	fopen, _Ropen	OPEN
Acquire	ACQ	ACQUIRE	QXXACQUIRE, _Racquire	Not supported
Get-attributes	POST	ACCEPT	QXXDEVATR, _Rdevatr	Not supported
Read	READ	READ	fread, _Rreadn	READ
Read-from- invited- program-devices	READ[1]	READ	QXXREADINVDEV followed by an fread, _Rreadindv	Not supported
Write	WRITE	WRITE	fwrite, _Rwrite	WRITE
Write/Read	EXFMT	Not supported	_Rwriterd	Not supported
Release	REL	DROP	QXXRELEASE, _Rrelease	Not supported

Figure A-2 (Page 2 of 2). High-Level Language I/O Operations

ICF Operation	ILE RPG/400 Operation Code	ILE COBOL/400 Procedure Statement	ILE C/400 Function[2]	ILE FORTRAN/400 Statement
Close	CLOSE	CLOSE	fclose, _Rclose	CLOSE

Note:

[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 you are using determines whether to issue an ICF read or read-from-invited-program-devices operation, based on the format of the read operation. For example, if a read is issued with a specific format or terminal specified, the read operation is interpreted as an ICF read operation. Refer to the appropriate language reference book for more information.

[2] ILE C/400 programming language is case sensitive.

DDS Keywords

Figure A-3 presents the data description specifications (DDS) keywords you can use to specify communications functions for SNUF. For a description of how to combine and use DDS keywords, see the book, *ICF Programming*.

Figure A-3. Valid DDS Keywords for SNUF

DDS Keyword	Function
ALWWRT	Allow-write
CANCEL	Cancel
CNLINVITE	Cancel-invite
DETACH[1]	Detach
ENDGRP	End-of-group
EOS	End-of-session
EVOKE[1]	Evoke
FAIL	Fail
FMH	Function management header
INVITE	Invite
NEGRSP	Negative-response
RCVCANCEL	Receive-cancel
RCVCONFIRM	Receive-confirm
RCVDETACH[1]	Receive-detach
RCVENDGRP	Receive-end-of-group
RCVFMH	Receive-function management header
RCVNEGRSP	Receive-negative-response
RCVTRNRND	Receive-turnaround
RECID	Record-identification
RQSWRT	Request-to-write
RSPCONFIRM	Respond-to-confirm
SECURITY	Security
TIMER	Timer
VARLEN	Variable-length data

Note:

[1] This function is not supported while running in half-duplex contention mode.

System-Supplied Formats

You can also use system-supplied communications formats to specify communications functions in your program. Figure A-4 shows the formats you can use with SNUF. For a description of how to use system-supplied formats, see the book, *ICF Programming*.

Figure A-4. Valid System-Supplied Formats for SNUF

System-Supplied Formats	Function
$$CANL	Cancel with invite
$$CANLNI	Cancel
$$CNLINV	Cancel-invite
$$EOS	End-of-session
$$EVOK[1]	Evoke with invite
$$EVOKET[1]	Evoke with detach
$$EVOKNI[1]	Evoke
$$FAIL	Fail
$$NRSP	Negative-response with invite
$$NRSPNI	Negative-response
$$POSRSP	Positive-response
$$RCD	Request-to-write with invite
$$SEND	Send with invite
$$SENDE	Send with end-of-group
$$SENDET[1]	Send with detach
$$SENDFM	Send FM Header with invite
$$SENDNF	Send FM Header
$$SENDNI	Send
$$TIMER	Timer

Note:

[1] This format is not supported while running in half-duplex contention mode.

Appendix B. Return Codes, Messages, and Sense Codes

Return Codes

This section describes all the return codes that are valid for SNUF. 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

Major Code 00 – Operation completed successfully.

Description: The operation issued by your program completed successfully. Your program may have sent or received some data, or may have received a message from the remote system.

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

Code	Description/Action
0000	**Description:** For input operations issued by your program, 0000 indicates that your program received some data with a turnaround indication. The remote program is ready to receive data. For output operations issued by your program, 0000 indicates that the last output operation completed successfully and that your program can continue to send data. **Action:** If your program received a turnaround on an input operation, issue an input or output operation. For the actions which can be taken after 0000 is received, refer to the following table:

Figure B-1. Actions for Return Code 0000

Type of Session	Last Operation Issued	Actions Your Program Can Take
Started by a source program	Acquire or open	Issue an evoke or timer function, or a get-attributes operation.
	Evoke with detach or write with detach	Issue another evoke function, issue a release operation, continue local processing, or end.
	Any other output operation	Issue another output operation (except evoke), or issue an input operation.
	End-of-Session	Continue local processing or end.
Started by a remote program start request[1]	Acquire or open	Issue an input or output operation.
	Write with detach	Continue local processing or end. This session has ended.
	Any other output operation	Issue another output operation (except evoke), or issue an input operation.
	End-of-Session	Continue local processing or end.

1 A target program (started by a program start request) cannot issue an evoke function in this session; it can issue an evoke function only in a different session that it has first acquired.

0001 **Description:** On a successful input operation, your program received some data. Your program must continue to receive data until it receives a turnaround indication (which allows your program to send data) or a detach indication.

Action: Issue another input operation. If your program detects a turnaround indication, it can issue an output operation.

0003 **Description:** On a successful input operation, your program received some data with an end-of-group indication. Your program may also have received a definite-response-request with the end-of-group indication.

Action: Issue an input operation to receive the next group of records. You may also issue a $$POSRSP format to respond to the definite-response-request. If you are using DDS keywords, you can use the RCVCONFIRM and RSPCONFIRM keywords. If your program receives a definite-response-request and you ignore it, SNUF sends a positive response to the application when the next input/output operation is performed.

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

Action: Issue a second input operation to receive the FMH data, then issue an output operation.

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

Action: Your program can issue a second input operation to receive the FMH data, then issue another input operation to continue receiving data until it receives a turnaround indication or a detach indication.

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

Action: Issue a second input operation to receive the FMH data, then issue another input operation to receive the next group of records.

0008 **Description:** On a successful input operation, your program received a detach indication with the last of the data. The communications transaction with the remote program has ended, but the session with the remote system is still active.

Action: If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the remote program started the transaction, your program can either issue an end-of-session function or end.

000C **Description:** On a successful input operation, your program received the last of the data with a function-management-header (FMH) and a detach indication. The communications transaction with the remote program has ended, but the session with the remote system is still active.

Action: Issue a second input operation to receive the data. If your program started the transaction, issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end your program. If a program start request from the remote program started the transaction, issue an end-of-session function or end your program.

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

Action: Issue an input operation as soon as possible.

0020 **Description:** On a successful input operation, your program received a remote system message and a turnaround indication. The message is in your program's input buffer and describes why the previous operation was rejected.

Action: Handle the message in the input buffer (for example, display it). Your program now has control of the session, and can issue an output operation.

0021 **Description:** On a successful input operation, your program received a remote system message which is now in your program's input buffer. Your program should continue to receive input.

Action: Handle the message in the input buffer (for example, display it), and issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation.

0023 **Description:** On a successful input operation, your program received a remote system message with an end-of-group indication. The system message is now in your input buffer.

Action: Handle the message in the input buffer (for example, display it), and issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation.

0025 **Description:** On a successful input operation, your program received a remote system message with a function-management-header (FMH).

Action: Process the FMH and issue a second input operation to receive the system message.

0027 **Description:** On a successful input operation, your program received a remote system message with a function-management-header (FMH) and an end-of-group indication.

Action: Process the FMH and issue a second input operation to receive the system message.

0028 **Description:** On a successful input operation, your program received a detach indication with a remote system message. The communications transaction with the remote program has ended, but the session with the remote system is still active. The system message is in your program's input buffer and describes the status of the transaction that has ended.

Action: Handle the message in the input buffer (for example, display it). If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the remote program started the transaction, your program can either issue an end-of-session function or end.

0030 **Description:** On a successful input operation, your program received a truncated remote system message and a turnaround indication. The message is in your program's input buffer, and was truncated because it was too long for the buffer. The message describes why the previous operation was rejected.

Action: Handle the message in the input buffer (for example, display it). Your program now has control of the session, and can issue an output operation.

0031 **Description:** On a successful input operation, your program received a truncated remote system message. The message is in your program's input buffer, and was truncated because it was too long for the buffer. Your program should continue to receive input.

Action: Handle the message in the input buffer (for example, display it), and issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation.

0033 **Description:** On a successful input operation, your program received a truncated system message with an end-of-group indication. The message is in your input buffer, and was truncated because it was too long for the buffer.

Action: Handle the message in the input buffer (for example, display it), and issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation.

0035 **Description:** On a successful input operation, your program received a truncated system message with a function-management-header (FMH). The message is in your input buffer, and was truncated because it was too long for the buffer.

Action: Process the FMH and issue a second input operation to receive the system message.

0037 **Description:** On a successful input operation, your program received a truncated system message with a function-management-header (FMH) and an end-of-group indication.

Action: Process the FMH and issue a second input operation to receive the system message.

0038 **Description:** On a successful input operation, your program received a detach indication with a truncated remote system message. The communications transaction with the remote program has ended, but the session with the remote system is still active. The message is in your program's input buffer and describes the status of the transaction that has ended. The message was truncated because it was too long for the buffer.

Action: Handle the message in the input buffer (for example, display it). If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the remote program started the transaction, your program can either issue an end-of-session function or end.

Major Code 02

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

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

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

Code	Description/Action
0200	**Description:** On a successful input operation, your program received some data with a turnaround indication. Also, your job is being ended (controlled). The remote program is ready to receive data from your program. **Action:** Your program can issue an input or output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
0201	**Description:** On a successful input operation, your program received some data. Also, your job is being ended (controlled). Your program can continue to receive data until it receives a turnaround indication (which allows your program to send data) or a detach indication. **Action:** Your program can issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
0203	**Description:** On a successful input operation, your program received some data with an end-of-group indication. Also, your job is being ended (controlled). **Action:** Your program can issue an input operation to receive the next group of records. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
0204	**Description:** On a successful input operation, your program received some data with a function-management-header (FMH) and a turnaround indication. Also, your job is being ended (controlled). The remote program is ready to receive data. **Action:** Your program can issue a second input operation to receive the FMH data, then issue an output operation. However, the recom-

mended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0205 **Description:** On a successful input operation, your program received some data with a function-management-header (FMH). Also, your job is being ended (controlled).

Action: Your program can issue a second input operation to receive the FMH data, then issue another input operation to continue receiving data until it receives a turnaround indication or a detach indication. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

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

Action: Your program can issue a second input operation to receive the FMH data, then issue another input operation to receive the next group of records. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0208 **Description:** On a successful input operation, your program received a detach indication with the last of the data. The communications transaction with the remote program has ended, but the session with the remote system is still active. Also, your job is being ended (controlled).

Action: If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the remote program started the transaction, your program can either issue an end-of-session function or end. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

020C **Description:** On a successful input operation, your program received the last of the data with a function-management-header (FMH) and a detach indication. The communications transaction with the remote program has ended, but the session with the remote system is still active. Also, your job is being ended (controlled).

Action: Issue a second input operation to receive the data. If your program started the transaction, issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end your program. If a program start request from the remote program started the transaction, issue an end-of-session function or end your program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0220 **Description:** On a successful input operation, your program received a remote system message and a turnaround indication. The message is in your program's input buffer and describes why the previous operation was rejected. Also, your job is being ended (controlled).

Action: Handle the message in the input buffer (for example, display it). Your program now has control of the session, and 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.

0221 **Description:** On a successful input operation, your program received a remote system message which is now in your program's input buffer. Also, your job is being ended (controlled). Your program should continue to receive input.

Action: Handle the message in the input buffer (for example, display it), and issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0223 **Description:** On a successful input operation, your program received a remote system message with an end-of-group indication. The system message is now in your input buffer. Also, your job is being ended (controlled).

Action: Handle the message in the input buffer (for example, display it), and issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0225 **Description:** On a successful input operation, your program received a remote system message with a function-management-header (FMH). Also, your job is being ended (controlled).

Action: Process the FMH and issue a second input operation to receive the system message. 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.

0227 **Description:** On a successful input operation, your program received a remote system message with a function-management-header (FMH) and an end-of-group indication. Also, your job is being ended (controlled).

Action: Process the FMH and issue a second input operation to receive the system message. 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.

0228 **Description:** On a successful input operation, your program received a detach indication with a remote system message. The communications transaction with the remote program has ended, but the session with the remote system is still active. The system message is in your program's input buffer and describes the status of the transaction that has ended. Also, your job is being ended (controlled).

Action: Handle the message in the input buffer (for example, display it). If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the remote program started the transaction, your program can either issue an end-of-session function or end. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0230 **Description:** On a successful input operation, your program received a truncated remote system message and a turnaround indication. The message is in your program's input buffer, and was truncated because it was too long for the buffer. The message describes why the previous operation was rejected. Also, your job is being ended (controlled).

Action: Handle the message in the input buffer (for example, display it). Your program now has control of the session, and 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.

0231 **Description:** On a successful input operation, your program received a truncated remote system message. The message is in your program's input buffer, and was truncated because it was too long for the buffer. Also, your job is being ended (controlled). Your program should continue to receive input.

Action: Handle the message in the input buffer (for example, display it), and issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0233 **Description:** On a successful input operation, your program received a truncated system message with an end-of-group indication. The message is in your input buffer, and was truncated because it was too long for the buffer. Also, your job is being ended (controlled).

Action: Handle the message in the input buffer (for example, display it), and issue another input operation. If your program detects the equivalent of a turnaround indication, it can issue an output operation. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

0235 **Description:** On a successful input operation, your program received a truncated system message with a function-management-header (FMH). The message is in your input buffer, and was truncated because it was too long for the buffer. Also, your job is being ended (controlled).

Action: Process the FMH and issue a second input operation to receive the system message. 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.

0237 **Description:** On a successful input operation, your program received a truncated system message with a function-management-header (FMH) and an end-of-group indication. Also, your job is being ended (controlled).

Action: Process the FMH and issue a second input operation to receive the system message. 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.

0238 **Description:** On a successful input operation, your program received a detach indication with a truncated remote system message. The communications transaction with the remote program has ended, but the session with the remote system is still active. The message is in your program's input buffer and describes the status of the transaction that has ended. The message was truncated because it was too long for the buffer. Also, your job is being ended (controlled).

Action: Handle the message in the input buffer (for example, display it). If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the remote program started the transaction, your program can either issue an end-of-session function or end. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

Major Code 03

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

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

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

Code	Description/Action
0300	**Description:** On a successful input operation, your program received a turnaround indication without any data. The session is still active. **Action:** Issue an input or output operation.
0301	**Description:** On a successful input operation, your program received no data. Your program must continue to receive input until it receives a turnaround or detach indication. **Action:** Issue an input operation.
0303	**Description:** On a successful input operation, your program received an end-of-group indication without any data. **Action:** Issue another input operation.
0308	**Description:** On a successful input operation, your program received a detach indication without any data. The communications transaction with the remote program has ended, but the session with the remote system is still active. If you specified the DDS keyword RCVDETACH, the receive-detach indicator is also set on. **Action:** If your program started the session, it can issue another evoke function (to start another program), issue a release operation (to perform local processing or to start another session), or end. If a program start request from the remote program started the transaction, your program can either issue an end-of-session function or end.
0309	**Description:** On a read-from-invited-program-devices operation, your program did not receive any data. Also, your job is being ended (controlled). **Action:** Your program can continue processing. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job. **Messages:** `CPF4741 (Notify)`
0310	**Description:** On a read-from-invited-program-devices operation, the time interval specified by a timer function in your program or by the WAITRCD value specified for the ICF file expired. **Action:** Issue the intended operation after the specified time interval has ended. For example, if you were using the time interval to control

the length of time to wait for data, you can issue another read-from-invited-program-devices operation to receive the data.

Note: Since no specific program device name is associated with the completion of this operation, the program device name in the common I/O feedback area is set to *N. Therefore, your program should not make any checks based on the program device name after receiving the 0310 return code.

Messages:

```
CPF4742 (Status)
CPF4743 (Status)
```

Major Code 04

Major Code 04 – Output exception occurred.

Description: An output exception occurred because your program attempted to send data when it should be receiving data. The data from your output operation was not sent. You can attempt to send the data later.

Action: Issue an input operation to receive the data.

Code	Description/Action

0412	**Description:** An output exception occurred because your program attempted to send data when it should be receiving data 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.
	
	Note: If your program issues another output operation before an input operation, your program receives a return code of 831C.
	
	Messages:
	
	`CPF4750 (Notify)`
	`CPF5076 (Notify)`

Major Codes 08 and 11

Major Codes 08 and 11 – Miscellaneous program errors occurred.

Description: The operation just attempted by your program was not successful. The operation may have failed because it was issued at the wrong time.

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

Code	Description/Action
0800	**Description:** The acquire operation just attempted by your program was not successful. Your program tried to acquire a program device that was already acquired and is still active. **Action:** If the session associated with the original acquire operation is the one needed, your program can begin communicating in that session since it is already available. If you want a different session, issue another acquire operation for the new session by specifying a different program device name in the PGMDEV parameter of the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command that precedes the program. **Messages:** CPD4077 (Diagnostic) CPF5041 (Status) CPF50A0 (Status)
1100	**Description:** The read-from-invited-program-devices operation just attempted by your program was not successful because your program tried this operation when no program devices were invited and no timer function was in effect. **Action:** Issue an invite function (or a combined operation that includes an invite) followed by a read-from-invited-program-devices operation. **Messages:** CPF4740 (Notify)

Major Code 34

Major Code 34 – Input exception occurred.

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

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

Code **Description/Action**

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	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)`
	`CPF5089 (Escape)`
	`CPF5257 (Escape)`
	`CPF5411 (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, output, or both.
- If your program tried selecting user buffers and blocked data together, it should try selecting one or the other, but not both.
- If your program tried to open a non-source file as a source file, either change the file name or change the library name.
- If there was a mismatch on the INDARA keyword, either correct the file or correct your program so that the two match.
- If no program devices were previously acquired for a shared file, acquire one or more program devices for the file.

Messages:

```
CPF4133 (Escape)
CPF4156 (Escape)
CPF4238 (Escape)
CPF4250 (Escape)
CPF4345 (Escape)
CPF5522 (Escape)
CPF5549 (Escape)
```

80ED **Description:** The open operation attempted by your program was not successful because there is a record format level mismatch between your program and the ICF file.

Action: Close the file. Compile your program again to match the file level of the ICF file, or change or override the file to LVLCHK(*NO); then open the file again.

Messages:

```
CPF4131 (Escape)
```

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

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

Messages:

```
CPF4104 (Escape)
```

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

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

Action:

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

Messages:

```
CPF4132 (Escape)
CPF5129 (Escape)
```

Major Code 81

Major Code 81 – Permanent session error (irrecoverable).

Description: An irrecoverable session error occurred during an I/O operation. Your session cannot continue and has ended. Before communications can resume, the session must be established again by using an acquire operation or another program start request. Recovery from this error is unlikely until the problem causing the error is detected and corrected. Operations directed to other sessions associated with the file should work.

Action: You can perform the following general actions for all 81xx return codes. Specific actions are given in each minor return code description.

If your program initiated the session, you can:

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

If your session was initiated by a program start request from the 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.

Code	Description/Action
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)`
	`CPF4542 (Escape)`
	`CPF5128 (Escape)`

8196	**Description:** The remote system sent a Systems Network Architecture (SNA) UNBIND command to your system, or the session was ended locally. **Action:** To start another session, issue the acquire operation again. Otherwise, your program can continue local processing or end. **Messages:** `CPF5165 (Escape)`
819D	**Description:** On an input or output operation, your program received some unexpected data from the remote program. The session has ended. **Action:** Your program is expecting the remote program to send a detach indication with no data. **Messages:** `CPF5089 (Escape)`
81B9	**Description:** On an input operation, the remote program sent a data record whose length was greater than the maximum user record length specified for your session. The session has ended. **Action:** Verify that the value in the user record length (RCDLEN) parameter on the device description (CRTDEVSNUF) command or on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command is large enough for the longest record to be received. **Messages:**

```
CPF5305 (Escape)
CPF5335 (Escape)
CPF5388 (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.

Code **Description/Action**

8209 **Description:** The open or acquire operation issued by your program was not successful because a prestart job is being canceled. One of the following may have occurred:

- An End Job (ENDJOB), End Prestart Job (ENDPJ), End Subsystem (ENDSBS), End System (ENDSYS), or Power Down System (PWRDWNSYS) command was being issued.
- The maximum number of prestart jobs (MAXJOBS parameter) was reduced by the Change Prestart Job Entry (CHGPJE) command.
- The value for the maximum number of program start requests allowed (specified in the MAXUSE parameter on the ADDPJE or CHGPJE command) was exceeded.
- Too many unused prestart jobs exist.
- The prestart job had an initialization error.

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

Messages:

```
CPF4292 (Escape)
CPF5313 (Escape)
```

8233 **Description:** A program device name that was not valid was detected. Either an ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command was not run, or the program device name in your program does not match the program device name specified in the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE 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:

```
CPF4121 (Escape)
CPF4168 (Escape)
CPF4182 (Escape)
```

```
CPF4369 (Escape)
CPF4370 (Escape)
CPF4375 (Escape)
CPF5257 (Escape)
CPF5274 (Escape)
CPF5317 (Escape)
CPF5318 (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)
```

8285 **Description:** On an open or acquire operation, the attempt by your program to call a remote location automatically using a switched connection was not successful. The number specified on the controller description was dialed, but the connection was not established. Possible causes are that the line was busy, that there was no answer, or that the number dialed was disconnected. The session was not started.

Action: Verify that the number you are dialing is correct and that the remote system is ready for the call. Also verify that the line description you are using is varied on and is included in the switched line list on the controller description. Your program can issue the open or acquire operation again, continue local processing, or end.

Messages:

```
CPF4179 (Escape)
CPF5260 (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:

```
CPF4146 (Escape)
```

```
CPF5353 (Escape)
```

82A1 **Description:** The acquire operation issued by your program was not successful because the logon portion of the acquire operation failed. The host subsystem may have been inactive, or the name of the remote application program specified in the application ID (APPID) parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command may have been incorrect. The session was not started.

Action: Verify that the name of the remote application program is specified correctly on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. If the program name is specified correctly, contact the remote location and request that the host system be started, then issue the acquire operation again. Otherwise, your program can continue local processing, wait to send the acquire operation again, or end.

Messages:

```
CPF4180 (Escape)
CPF4226 (Escape)
CPF4758 (Escape)
CPF5228 (Escape)
CPF5516 (Escape)
CPF5518 (Escape)
```

82A5 **Description:** On an unsuccessful acquire operation, SNUF detected a combination of parameter values on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command that was not valid. The session was not started.

Action: Change the parameter values on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, then issue the acquire operation again.

Messages:

```
CPF4303 (Escape)
CPF5511 (Escape)
CPF5338 (Escape)
```

82A6 **Description:** On the open or acquire operation attempted by your program, a negative-response with sense data was received when the Systems Network Architecture (SNA) BIND command was sent to the user to start the session. The session was not started.

Action: Close the file. Check for an error in the format of the incorrect BIND command, or contact the remote system to determine why the command failed. After correcting the error, your program can issue the acquire operation again to start the session.

Messages:

```
CPF4227 (Escape)
CPF4333 (Escape)
CPF4527 (Escape)
CPF5517 (Escape)
CPF5538 (Escape)
```

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:

```
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 job was started by a program start request with the *REQUESTER device detached.
- The *REQUESTER device was released because an end-of-session was requested.
- The job does not have a *REQUESTER device; that is, the job was not started by a program start request.
- A permanent error occurred on the session.

Action:

- If the *REQUESTER device is already acquired and your program expects to communicate with the *REQUESTER device, use the program device that acquired the *REQUESTER.
- If the *REQUESTER device is not available and your program expects to communicate with the *REQUESTER device, the remote program must send a program start request without a detach function.
- If your program released its *REQUESTER device, before trying to acquire it, correct the error that caused your program
- 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)
```

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)
```

82B4 **Description:** The acquire operation issued by your program was not successful because a System/36 application program in the System/36 environment attempted to acquire an ICF program device for 3270 SNA program interface tasks.

Action: 3270 SNA program interface does not support System/36 application programs. Use an AS/400 program that is not running in the System/36 environment.

Messages:

```
CPF4233 (Escape)
CPF5336 (Escape)
```

82B5 **Description:** The acquire operation issued by your program was not successful because a 3270 SNA program interface session cannot use a SNUF device from an earlier release.

Action: Delete the SNUF device description, create a new device description, and issue the acquire operation again.

Messages:

```
CPF4136 (Escape)
```

82BB **Description:** The acquire operation issued by your program was not successful because the program device specified by your program was reserved for a program start request from the host. The session was not started.

Action: Specify a device which was not created as being capable of a program start request, or create this device again with the correct attributes.

Messages:

```
CPF4109 (Escape)
CPF5355 (Escape)
```

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

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

Messages:

```
CPF4348 (Escape)
CPF5521 (Escape)
```

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

Action: Your program can continue local processing or end. Verify that the name of the remote location is specified correctly in the 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)
```

82F5 **Description:** The open or acquire operation was not successful because your program tried to use a format selection option of *RMTFMT in the FMTSLT parameter on the ADDICFDEVE or OVRICFDEVE command. The session was not started.

Action: Change the value in the FMTSLT parameter on the ADDICFDEVE or OVRICFDEVE command, then issue the open or acquire operation again.

Messages:

```
CPF4347 (Escape)
CPF5515 (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 ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

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

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

Code	Description/Action
830B	**Description:** Your program attempted an operation that was not valid because the session was not yet acquired or has ended. The session may have ended because of a release operation, an end-of-session function, or a permanent error. Your program may have incorrectly handled a previous error. **Action:** Verify that your program does not attempt any operations without an active session. Also verify that your program correctly handles the permanent error or session-not-acquired return codes (80xx, 81xx, 82xx) it received on previously issued input and output operations. To recover from an incorrectly handled error condition, your program may or may not be able to issue another acquire operation, depending on the return code.

Messages:

```
CPD4079 (Diagnostic)
CPF4739 (Status)
CPF5067 (Escape)
CPF5068 (Escape)
CPF5070 (Escape)
```

830C **Description:** Your program received a function-management-header (FMH) that was greater than the request unit (RU) size.

Action: Verify that the host system actually sent the function-management-header. If the host system sent only data and no FMH, SNUF may have treated a data byte as the FMH length.

Messages:

```
CPF4770 (Notify)
```

830D **Description:** Your program received a shutdown indication from the host system. The host system has begun shutdown procedures, although the session is still usable.

Action: Finish activity for the session and release the session as soon as possible so the host system can complete shutdown procedures.

Messages:

```
CPF4771 (Notify)
```

8311 **Description:** Your program attempted an output operation while a message containing sense data was waiting to be received.

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

Messages:

```
CPF4772 (Notify)
```

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

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

Messages:

```
CPF4773 (Notify)
```

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

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

Messages:

```
CPF4774 (Notify)
```

831C **Description:** Your program's previous output operation received a return code of 0412, indicating that your program must receive information sent by the 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)
CPF5076 (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 SNUF.
- Your program requested a combination of operations or keywords that was not valid, such as a combined write-then-read operation with the invite function specified.
- Your program issued an input operation, or an output operation with the invite or allow-write function, for a file that was opened for output only.
- Your program issued an output operation for a file that was opened for input only.
- Your program issued a close operation with a temporary close option.

Action: Your program can try a different operation, issue a release operation or end-of-session function, or end. Correct the error in your program before trying to communicate with the 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 or allow-write function on an output operation. If such an operation is needed, then release the session, close the ICF file, and open the file again for input and output.

Messages:

```
CPF4564 (Escape)
CPF4764 (Notify)
CPF4766 (Notify)
CPF4790 (Notify)
CPF5132 (Escape)
CPF5149 (Escape)
```

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

- On an output operation, your program tried to send a data record that was longer than the MAXRCDLEN value specified for the ICF file.
- The program used a read or write operation that specified a data length greater than the record format in the ICF file.

- If this was a timer function, the format of the timer interval was not HHMMSS.
- If a system-defined format was used to specify the operation, or if the variable-length-data-record (VARLEN) function was used, then the length of the user buffer was not valid.
- For an SNA 3270 program interface application, the output buffer length was less than the length of the required header plus data; or, for the SSCP-LU record, the length was greater than the maximum allowed.

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)
CPF4786 (Notify)
CPF4787 (Notify)
CPF4825 (Notify)
CPF4827 (Notify)
```

8322 **Description:** Your program tried to issue a request-to-turnaround-with-invite function, a negative-response function, or a release operation. However, these are not valid while your program is in send state.

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

Messages:

```
CPF4775 (Notify)
```

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

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

Messages:

`CPF4776 (Notify)`

8324 **Description:** On an output operation, your program tried to send a function-management-header (FMH) with a record that was not the first record in the chain. An FMH is valid only at the beginning of a chain; it is not valid within the chain. The session is still active.

Action: Change your program so it sends the FMH with the first record in a chain.

Messages:

`CPF4778 (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 invalid operation.

Messages:

`CPF4779 (Notify)`

8327 **Description:** The input or output operation issued by your program was not successful because there was no active transaction. Either the transaction has ended, or the transaction was never started.

Action: If your program wants to start a transaction, it can issue an evoke function. Otherwise, it can issue an end-of-session function or end. If a coding error in your program caused the error, correct your program.

Messages:

`CPF5098 (Notify)`

8329 **Description:** An evoke function that was not valid was detected in this session. Your program was started by a program start request and, therefore, cannot issue any evoke functions in this session.

Action: To recover, your program can try a different operation or function. To issue an evoke function in a different session, first issue an acquire operation (using a different program device name), then try the evoke function. Otherwise, your program can issue an end-of-session function, continue local processing, or end. If a coding error caused your program to attempt an evoke that was not valid, correct your program.

Messages:

`CPF5099 (Notify)`

832C **Description:** A release operation following an invite function was detected. Because your program issued the invite function, it cannot issue a release operation to end the invited session.

Action: Issue an input operation to satisfy the invite function, or issue a cancel-invite function to cancel the invite function; then try the release operation again. Otherwise, issue an end-of-session function to end the session. If a coding error caused your program to attempt a release operation that was not valid, correct your program.

Messages:

```
CPF4769 (Notify)
```

832D **Description:** Following an invite function, your program issued a request-to-write indication, a negative-response indication, a cancel reply, or an additional invite function. This operation failed because the original invite function must first be satisfied by an input operation.

Action: Issue an input operation to receive the data that was invited. Otherwise, issue an end-of-session function to end the session. If a coding error caused your program to attempt a request-to-write indication or an additional invite function, correct your program.

Messages:

```
CPF4924 (Notify)
```

832F **Description:** The evoke function or release operation issued by your program was not successful because your program attempted the operation while the current transaction was still active. The operation was not performed, but the session is still active.

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

Messages:

```
CPF4780 (Notify)
CPF4781 (Notify)
```

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

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

Messages:

```
CPF4782 (Notify)
```

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

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

Messages:

```
CPF4783 (Notify)
```

8332 **Description:** On an input operation, your program received a cancel function with a detach indication. The remote program sent some data and then canceled the transaction, possibly because it detected an error in the data. The session is still active.

Action: Normally, your program should discard the canceled data it received from the remote program, as the data may be in error. If your program started the session, it can issue an evoke function to start another transaction, issue an end-of-session function to continue local processing, or end. If your program was started by a program start request from the remote program, it can continue local processing or end.

Messages:

```
CPF4784 (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:

```
CPF4785 (Notify)
```

83B7 **Description:** On a previous operation, your program received an indicator that the SNA 3270 program interface command code or data flow indicator in the common header is not valid, or that the data following the common header is not valid.

Action: Correct the logic in your program that produced the error condition in the common header or the data, then try the operation again.

Messages:

```
CPF4788 (Notify)
CPF4789 (Notify)
CPF4795 (Notify)
```

83B8 **Description:** On a previous operation, your program received a CLEAR or an UNBIND request with a BIND forthcoming. The host system has reset the session. Data sent to or received from the remote system may have been lost. The session is now in contention state.

Action: Your program can continue with another input or output operation, or it can release the program device and close the ICF file.

Messages:

```
CPF5163 (Notify)
```

83E0 **Description:** Your program attempted an operation using a record format that was not defined for the ICF file.

Action: Verify that the name of the record format in your program is correct, then check to see whether the record format is defined in the file definition.

Messages:

```
CPF5054 (Notify)
```

83E8 **Description:** Your program attempted to issue a cancel-invite function to a session that was not invited. One of the following may have occurred:

- The invite function was implicitly canceled earlier in your program by a valid output operation.
- The invite function was satisfied earlier in your program by a valid input operation.
- Your program had already canceled the invite function, then tried to cancel it again.
- Your program never invited the session.

The session is still active.

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

Messages:

```
CPF4763 (Notify)
```

83F8 **Description:** Your program attempted to issue an operation to a program device that is marked in error due to a previous I/O or acquire operation. Your program may have handled the error incorrectly.

Action: Release the program device, correct the previous error, then acquire the program device again.

Messages:

```
CPF5293 (Escape)
```

Failed Program Start Requests

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

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

The following lists shows reason codes for failed program start requests.

Code	Reason Code Description
401	Program start request is received to a device that is not allocated to an active subsystem.
402	Requested device is currently being held by a Hold Communications Device (HLDCMNDEV) command.
403	User profile is not accessible.
404	Job description is not accessible.
405	Output queue is not accessible.
406	Maximum number of jobs defined by subsystem description are already active.
407	Maximum number of jobs defined by communications entry are already active.
408	Maximum number of jobs defined by routing entry are already active.
409	Library on library list is exclusively in use by another job.
410	Group profile cannot be accessed.
411	Insufficient storage in machine pool to start job.
412	System value not accessible.
501	Job description was not found.
502	Output queue was not found.
503	Class was not found.
504	Library on initial library list was not found.
505	Job description or job description library is damaged.
506	Library on library list is destroyed.
507	Duplicate libraries were found on library list.
508	Storage-pool defined size is zero.
602	Transaction program-name value is reserved but not supported.
604	Matching routing entry was not found.
605	Program was not found.
704	Password is not valid.
705	User is not authorized to device.
706	User is not authorized to subsystem description.
707	User is not authorized to job description.
708	User is not authorized to output queue.
709	User is not authorized to program.
710	User is not authorized to class.
711	User is not authorized to library on library list.
712	User is not authorized to group profile.
713	User ID is not valid.
714	Default user profile is not valid.
715	Neither password nor user ID was provided, and no default user profile was specified in the communications entry.
718	No user ID.
722	A user ID was received, but a password was not sent.
723	No password was associated with the user ID.
725	User ID does not follow naming convention.
726	User profile is disabled.
801	Program initialization parameters are present but not allowed.
802	Program initialization parameter exceeds 2000 bytes.
803	Subsystem is ending.
804	Prestart job is inactive or is ending.
805	WAIT(NO) was specified on the prestart job entry, and no prestart job was available.
806	The maximum number of prestart jobs that can be active on a prestart job entry was exceeded.

807	Prestart job that ended when a program start request was being received.
901	Program initialization parameters are not valid.
902	Number of parameters for program not valid.
903	Program initialization parameters are required but not present.
1001	System logic error. Notification that an unexpected condition has stopped the running of a program or an unexpected return code encountered.
1002	System logic error. Notification that an unexpected condition has stopped the running of a program or an unexpected return code encountered while receiving program initialization parameters.
1501	Character in procedure name not valid.
1502	Procedure is not found.
1503	System/36 environment library is not found.
1504	Library QSSP is not found.
1505	File QS36PRC is not found in library QSSP.
1506	Procedure or library name is greater than 8 characters.
1507	Current library is not found.
1508	Not authorized to current library.
1509	Not authorized to QS36PRC in current library.
1510	Not authorized to procedure in current library.
1511	Not authorized to System/36 environment library.
1512	Not authorized to file QS36PRC in System/36 environment library.
1513	Not authorized to procedure in System/36 environment library.
1514	Not authorized in library QSSP.
1515	Not authorized to file QS36PRC in QSSP.
1516	Not authorized to procedure in QS36PRC in QSSP.
1517	Unexpected return code from System/36 environment support.
1518	Problem phase program is not found in QSSP.
1519	Not authorized to problem phase program in QSSP.
1520	Maximum number of target programs started (100 per System/36 environment).
2111	Program name is missing or not valid.
2118	Function-management-header not valid.
2123	End bracket or end chain missing.

Appendix C. Considerations for Host System Programmers

This appendix contains information intended for CICS/VS or IMS/VS host system application programmers. It also contains information for System/370, System/390, 33xx, and 43xx system programmers. Generally, the AS/400 programmer does not need the information in this section. However, when SNA upline facility (SNUF) is configured, the AS/400 programmer needs to know certain parameter values specified during host system generation. For SNA 3270 program interface host considerations, see Appendix D.

Generating the Host System With VTAM/NCP

You must complete host system generation before you begin communications with the AS/400 system. With VTAM/NCP, you can generate the host system with CICS/VS, IMS/VS, or both, to communicate with the AS/400 system in the network. See the specific creation considerations for the host systems in “Programming for CICS/VS Systems” on page C-5 and “Programming for IMS/VS Systems” on page C-8.

Defining Physical Unit Parameters

You must define the AS/400 system during Virtual Communications Access Method/Network Control Program (VTAM/NCP) generation. Each AS/400 controller description is represented as a physical unit in VTAM. Therefore, each AS/400 controller description that SNUF uses requires a physical unit definition.

Note: The following parameters on the physical unit definition pertain to a SNUF configuration created for a synchronous data link control (SDLC) line description. Slight differences may be required for token-ring network, Ethernet, and X.25 line descriptions.

PUTYPE parameter = 2
The physical unit type must be 2.

ADDR parameter = xx
This parameter specifies the station address. This parameter must be the same as the station address specified in the line description (CRTLINSDLC command) or controller description (CRTCTLHOST command).

ISTATUS parameter = ACTIVE / INACTIVE
This parameter specifies if the physical unit should be turned on when its major node is turned on.

MAXDATA parameter = 521
This parameter specifies the maximum amount of data the AS/400 system can receive, and must match the value specified for the MAXFRAME parameter on the AS/400 system. The default value shown here (521) defines a 512-byte buffer plus 9 bytes for the transmission header and the request/response header. Other acceptable frame sizes (that match a valid MAXFRAME value) are: 265, 1033, 1466, 1994, and 2057.

MAXOUT parameter = 7
This parameter specifies the number of frames that Network Control Program (NCP) sends to the AS/400 system before waiting for a response. For best performance, specify 7.

DISCNT parameter = YES / NO
This parameter specifies if VTAM disconnects the physical unit when the last logical unit session is ended. DISCNT=NO allows the AS/400 system to remain active when no sessions are active. VTAM turns off the physical unit when the last SNUF application on the line is turned off. DISCNT=YES disconnects the AS/400 system when the last session ends. SNUF remains active until a deactivate operation is performed. DISCNT=YES also causes VTAM to ignore the deactivate request. If switched lines and more than one location are configured, specify DISCNT=YES.

IDBLK parameter = 056 and IDNUM Parameter = number
These parameters make up the exchange identifier. Specify these parameters for a switched line only. For the AS/400 system, you must specify IDBLK as 056. The IDNUM must be the same as the EXCHID parameter specified in the line description (CRTLINSDLC command).

SSCPFM parameter = USSSCS
Specify SSCPFM=USSSCS to indicate that the AS/400 logical units associated with this physical unit use character-coded messages to communicate with VTAM. The AS/400 system requires character-coded messages.

USSTAB parameter = name
This parameter specifies the name of an unformatted systems services (USS) definition table. Since SNUF requires the IBM*-supplied definition table, do not specify this parameter.

Defining Logical Unit Parameters

Each SNUF session corresponds to an Systems Network Architecture (SNA) logical unit and requires a logical unit definition in the VTAM creation. The following parameters on the logical unit definition apply to SNUF.

LOCADDR parameter = address
This parameter specifies the local address of the session, which is equivalent to a logical unit number. You can define up to 255 logical units.

ENCR parameter = NONE
This parameter specifies the type of encryption to be used. The AS/400 system does not support encryption for SNUF, so specify ENCR=NONE.

ISTATUS parameter = ACTIVE / INACTIVE
This parameter specifies if the logical unit is to be turned on when the physical unit is turned on.

PACING parameter = count
This parameter specifies how timing control is handled between NCP and the logical unit. Pacing controls the rate of data flow between the AS/400 program and the host system. It allows the receiver to control the rate at which the sender sends requests. If timing control is necessary, the recommended timing control value is 7.

Sending the VTAM BIND Command

VTAM sends the BIND command to SNUF. A session is not started unless a correctly formatted BIND is received by the AS/400 system. The parameters for the BIND command for CICS/VS and IMS/VS systems are described in Figure C-1.

Figure C-1 (Page 1 of 2). Parameters for the BIND Command

Byte (Decimal)	Value (Hex)	Meaning
0	31	SNA BIND request code
1	01	Format (nonnegotiable)
2	03, 04	Function management profile
3[1]	03, 04	Transmission services profile
4	90, 91	Multiple request units—exception response
	A0, A1	Multiple request units—definite response
	B0, B1	Multiple request units—definite or exception response
5	90, 91	Multiple request units—exception response
	A0, A1	Multiple request units—definite response
	B0, B1	Multiple request units—definite or exception response
6, 7	2080	Common protocol
	3080	Common protocol
	6080	Common protocol
	7080	Common protocol
	0040[2]	Common protocol
	4040[2]	Common protocol
8, 9[3]		Transmission services use
10[4]	85	Maximum request/response unit size (sec to pri)

Figure C-1 (Page 2 of 2). Parameters for the BIND Command

Byte (Decimal)	Value (Hex)	Meaning
11[4]	85	Maximum request/response unit size (pri to sec)

Notes:

1 Transmission services profile 03 is not supported if message protection is requested.

2 Protocol is not supported if running in half-duplex contention mode.

3 SNUF does not check these bytes.

4 The value of these two bytes represent a coded hexadecimal size for request units. Each byte is in the form ab, where size = (a x 2^b). Values can be 00 through FB (inclusive). If the maximum request unit is not specified, a coded size of 89 (4096 byte) is assumed.

For a complete description of the BIND command, see the *VTAM Programming Guide*.

Example VTAM/NCP Generation

Figure C-2 on page C-4 is an example VTAM/NCP definition of physical and logical units on a nonswitched SDLC line. Figure C-3 on page C-5 is an example definition for units on a switched line. The parameters that correspond to SNUF configuration on the AS/400 system are highlighted and listed after each figure.

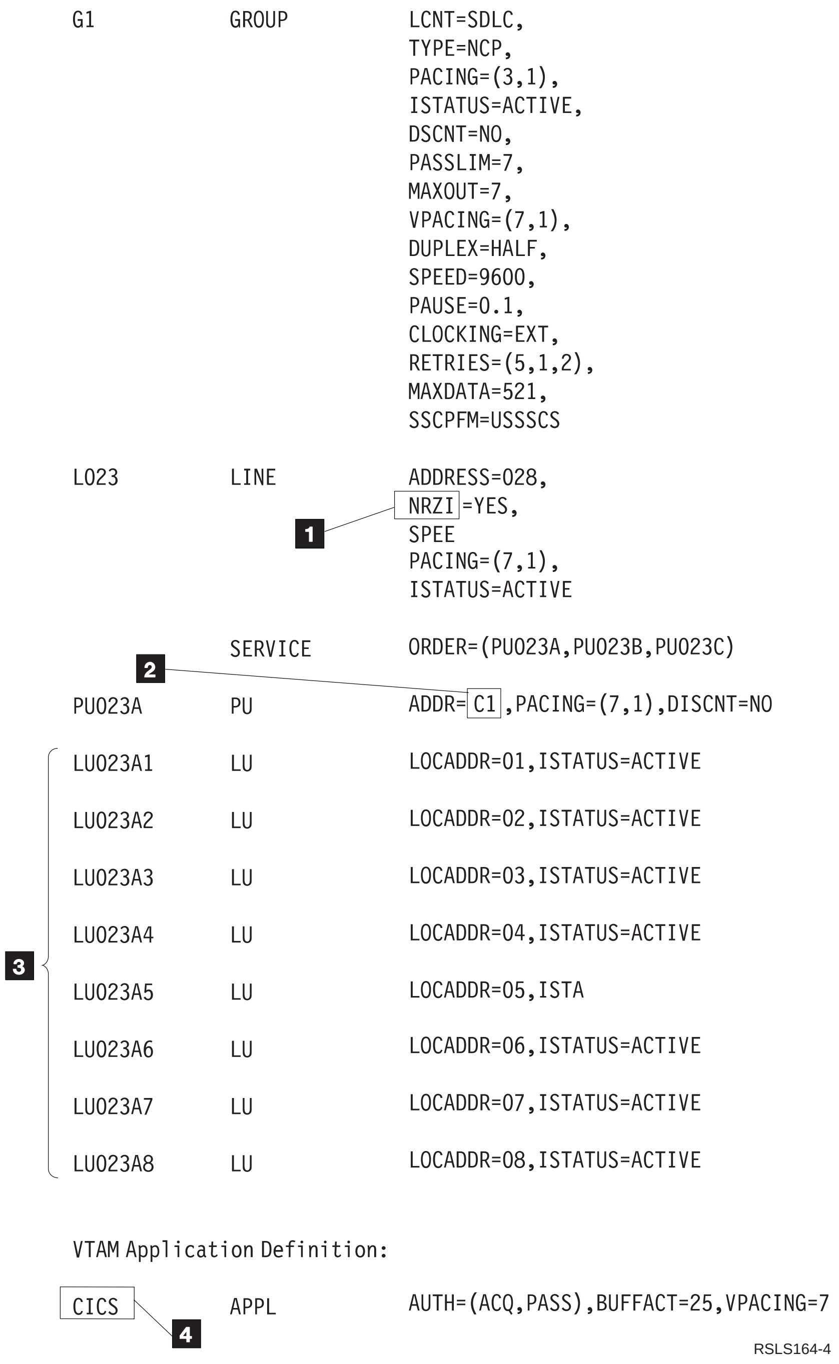
In Figure C-2 on page C-4:

1 This value corresponds to the NRZI parameter on the CRTLINSDLC command. The NRZI setting (Yes or No) must be the same for both the AS/400 system and the host system.

2 This value corresponds to the STNADR value on the CRTCTLHOST command.

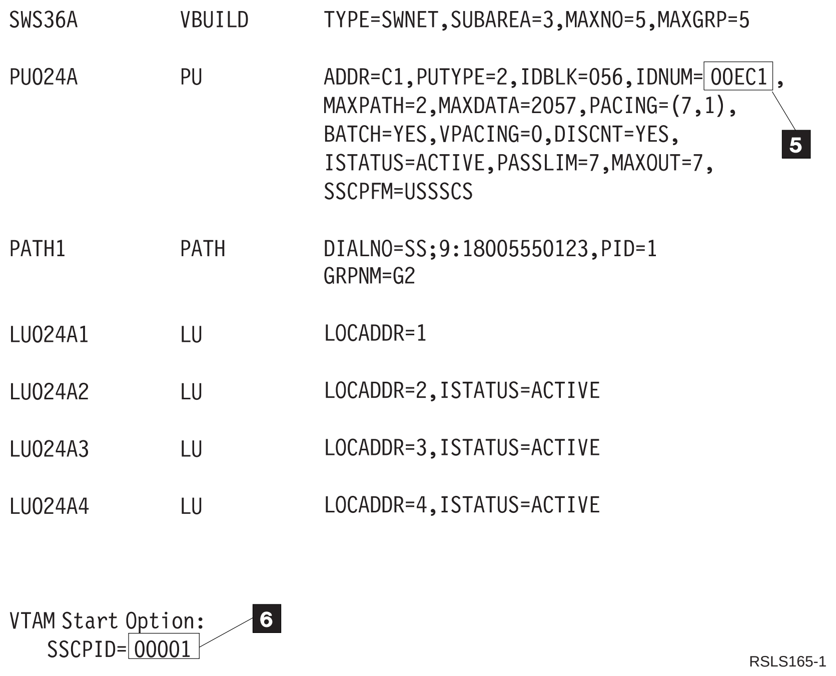
3 An active logical unit corresponds to a LOCADR specified by the CRTDEVSNUF command on the AS/400 system.

4 This name corresponds to the APPID parameter on the CRTDEVSNUF and OVRICFDEVE commands.



```
G1          GROUP          LCNT=SDLC,
                           TYPE=NCP,
                           PACING=(3,1),
                           ISTATUS=ACTIVE,
                           DSCNT=NO,
                           PASSLIM=7,
                           MAXOUT=7,
                           VPACING=(7,1),
                           DUPLEX=HALF,
                           SPEED=9600,
                           PAUSE=0.1,
                           CLOCKING=EXT,
                           RETRIES=(5,1,2),
                           MAXDATA=521,
                           SSCPFM=USSSCS

L023        LINE           ADDRESS=028,
                           NRZI=YES,
                           SPEE
                           PACING=(7,1),
                           ISTATUS=ACTIVE

            SERVICE        ORDER=(PU023A,PU023B,PU023C)

PU023A      PU             ADDR=C1,PACING=(7,1),DISCNT=NO

LU023A1     LU             LOCADDR=01,ISTATUS=ACTIVE

LU023A2     LU             LOCADDR=02,ISTATUS=ACTIVE

LU023A3     LU             LOCADDR=03,ISTATUS=ACTIVE

LU023A4     LU             LOCADDR=04,ISTATUS=ACTIVE

LU023A5     LU             LOCADDR=05,ISTA

LU023A6     LU             LOCADDR=06,ISTATUS=ACTIVE

LU023A7     LU             LOCADDR=07,ISTATUS=ACTIVE

LU023A8     LU             LOCADDR=08,ISTATUS=ACTIVE


VTAM Application Definition:

CICS        APPL           AUTH=(ACQ,PASS),BUFFACT=25,VPACING=7
```

Figure C-2. VTAM Creation on a Nonswitched SDLC Line



```
SWS36A      VBUILD      TYPE=SWNET,SUBAREA=3,MAXNO=5,MAXGRP=5

PU024A      PU          ADDR=C1,PUTYPE=2,IDBLK=056,IDNUM=00EC1,
                        MAXPATH=2,MAXDATA=2057,PACING=(7,1),
                        BATCH=YES,VPACING=0,DISCNT=YES,
                        ISTATUS=ACTIVE,PASSLIM=7,MAXOUT=7,
                        SSCPFM=USSSCS

PATH1       PATH        DIALNO=SS;9:18005550123,PID=1
                        GRPNM=G2

LU024A1     LU          LOCADDR=1

LU024A2     LU          LOCADDR=2,ISTATUS=ACTIVE

LU024A3     LU          LOCADDR=3,ISTATUS=ACTIVE

LU024A4     LU          LOCADDR=4,ISTATUS=ACTIVE


VTAM Start Option:
   SSCPID=00001
```

Figure C-3. VTAM Creation on a Switched SDLC Line

In Figure C-3:

5 This value corresponds to EXCHID on the CRTLINSDLC command.

6 This value corresponds to SSCPID on the CRTCTLHOST command. On the AS/400 system, this value is given in hexadecimal notation, and on the host system, it is given in decimal notation.

Performance Considerations

Give attention to the following host configuration parameters when defining the physical and logical units:

- The MAXDATA value should equal the AS/400 system MAXFRAME parameter value.
- The MAXOUT value should be as 7.
- A PACING value of zero (0) may be used. If this results in extra transmissions due to errors, then use a value between 1 and 63. The recommended value is 7.

Give attention to the following host parameters when defining Group and Build macros:

- The BFRS parameter is specified on the Group macro.
- The PASSLIM parameter is specified on the Build macro.

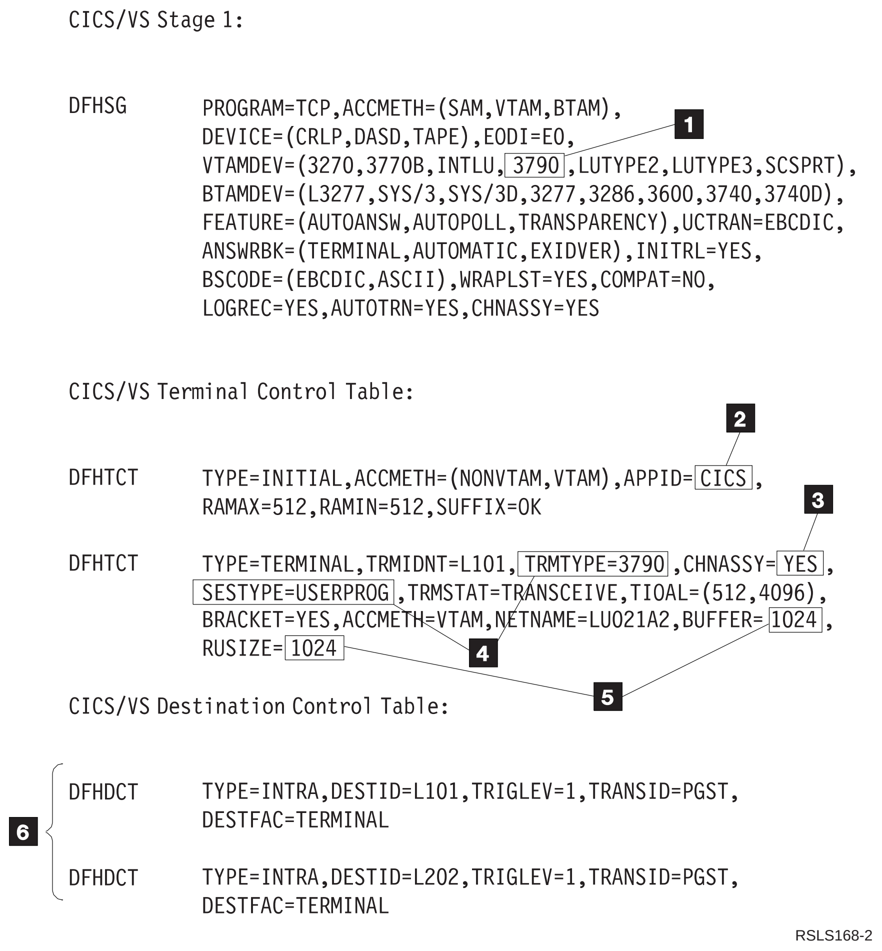
See the *Network Control Program: Resource Definition Reference* for additional information.

Programming for CICS/VS Systems

The CICS/VS system programmer defines SNUF logical units by coding SESTYPE=USERPROG and TRMTYPE=3790 in the DFHTCT TYPE=TERM macro-instruction.

To use function management headers in the application programs, specify the appropriate value for the INBFMH parameter in the program control table.

You should define the SNUF logical units in the CICS/VS terminal control table as 3790 full-function logical units. Figure C-4 on page C-6 shows examples of the table entries created by the DFHTCT macro for SNUF logical units. The highlighted parameters are listed and described after the figure.



```
CICS/VS Stage 1:


DFHSG          PROGRAM=TCP,ACCMETH=(SAM,VTAM,BTAM),
               DEVICE=(CRLP,DASD,TAPE),EODI=E0,
               VTAMDEV=(3270,3770B,INTLU,3790,LUTYPE2,LUTYPE3,SCSPRT),
               BTAMDEV=(L3277,SYS/3,SYS/3D,3277,3286,3600,3740,3740D),
               FEATURE=(AUTOANSW,AUTOPOLL,TRANSPARENCY),UCTRAN=EBCDIC,
               ANSWRBK=(TERMINAL,AUTOMATIC,EXIDVER),INITRL=YES,
               BSCODE=(EBCDIC,ASCII),WRAPLST=YES,COMPAT=NO,
               LOGREC=YES,AUTOTRN=YES,CHNASSY=YES


CICS/VS Terminal Control Table:


DFHTCT         TYPE=INITIAL,ACCMETH=(NONVTAM,VTAM),APPID=CICS,
               RAMAX=512,RAMIN=512,SUFFIX=OK

DFHTCT         TYPE=TERMINAL,TRMIDNT=L101,TRMTYPE=3790,CHNASSY=YES,
               SESTYPE=USERPROG,TRMSTAT=TRANSCEIVE,TIOAL=(512,4096),
               BRACKET=YES,ACCMETH=VTAM,NETNAME=LU021A2,BUFFER=1024,
               RUSIZE=1024

CICS/VS Destination Control Table:


DFHDCT         TYPE=INTRA,DESTID=L101,TRIGLEV=1,TRANSID=PGST,
               DESTFAC=TERMINAL

DFHDCT         TYPE=INTRA,DESTID=L202,TRIGLEV=1,TRANSID=PGST,
               DESTFAC=TERMINAL
```

Figure C-4. Sample CICS/VS Table Entries

In Figure C-4:

1 SNUF sessions are defined to CICS/VS as VTAM 3790 devices.

2 The APPID value on the CRTDEVSNUF or OVRICFDEVE commands must match the INITIAL DFCTCT macro identifier.

3 CHNASSY=YES allows the CICS/VS program to receive records exactly as the AS/400 program does with BATCH(*NO) on the ADDICFDEVE or OVRICFDEVE command. CHNASSY=NO matches BATCH(*YES) record handling on the AS/400 system.

4 TRMTYPE=3790 and SESTYPE=USERPROG define the session protocol to be used.

5 For best performance, the RUSIZE and BUFFER values should be greater than or equal to the RCDLEN value on the CRTDEVSNUF or OVRICFDEVE commands.

6 There are two methods to send program start requests to the AS/400 system:

- Define intrapartition destinations in the CICS/VS Destination Control Table
- Use the interval control START command, which does not require these CICS/VS table definitions.

End-of-Transaction Considerations

The evoke-with-detach and write-with-detach operations indicate that the AS/400 program no longer expects to communicate with the CICS/VS program that was started. To perform the detach function, SNUF sends an end-bracket indicator to the CICS/VS program if it is allowed to send the end-bracket indicator. If SNUF is not allowed to send the end-bracket, it sends a turnaround indicator to the CICS/VS program. In return, SNUF expects an end-bracket indicator without data from

CICS/VS. If SNUF does not receive the end-bracket indicator or if the indicator is accompanied by data, it abnormally ends the session. The CICS/VS program controls the end-bracket indicator by using the LAST parameter on the EXEC-CICS SEND command or the DFHTCT TYPE=WRITE macro instruction.

Program-Start-Request Considerations

SNUF accepts program start requests only on sessions reserved for the requests. Logical units reserved for program start requests must be started from the host system using the VARY command, the LOGAPPL parameter in the VTAM definition, or the CICS/VS ACQ master terminal command.

Use one of the following methods to send program start requests to an AS/400 system from CICS/VS:

- Transient data put to a transient data destination
- Interval control START command

Example CICS/VS Remote Program Start Request: The following examples show how a CICS/VS program starts an AS/400 application program. The example in Figure C-5 uses transient data put. The example in Figure C-6 on page C-8 uses the interval control START command.

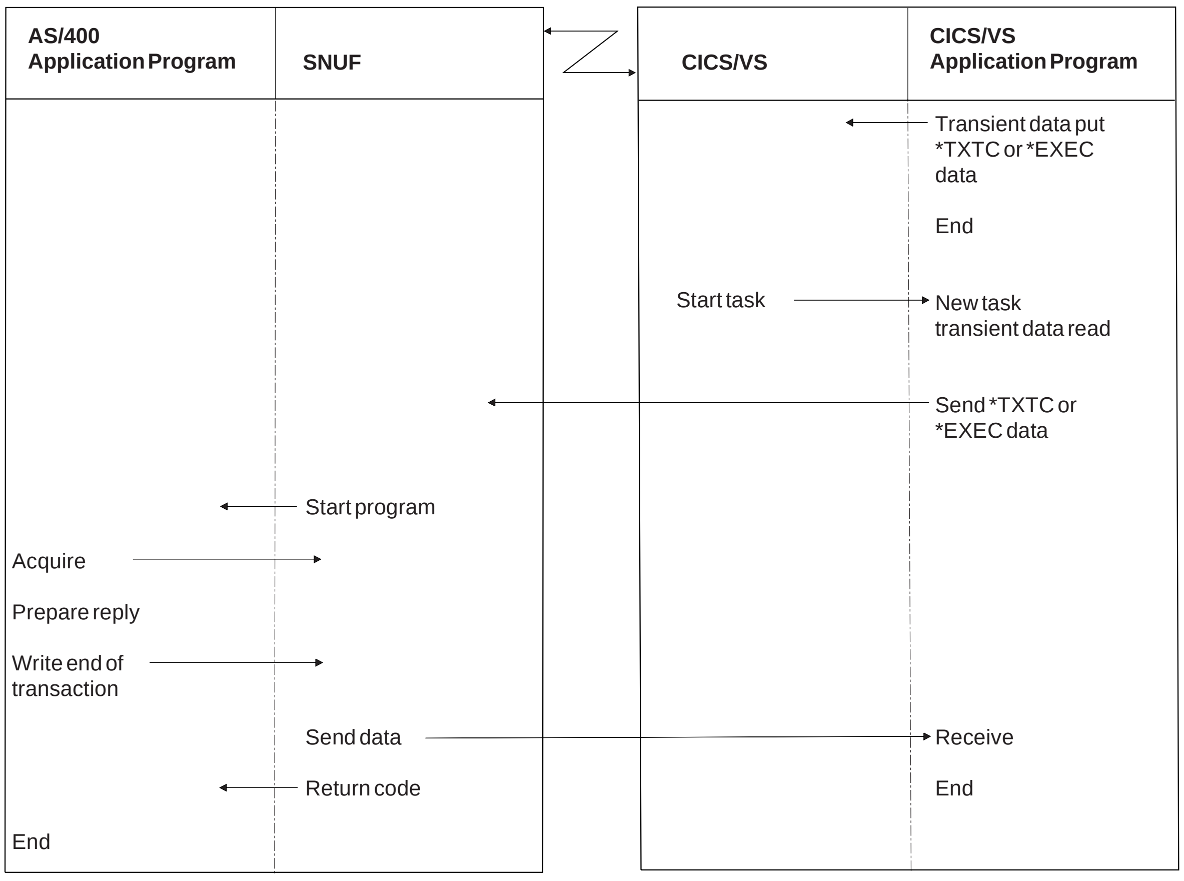


Figure C-5. Program Start Request Using the Transient Data Put Operation

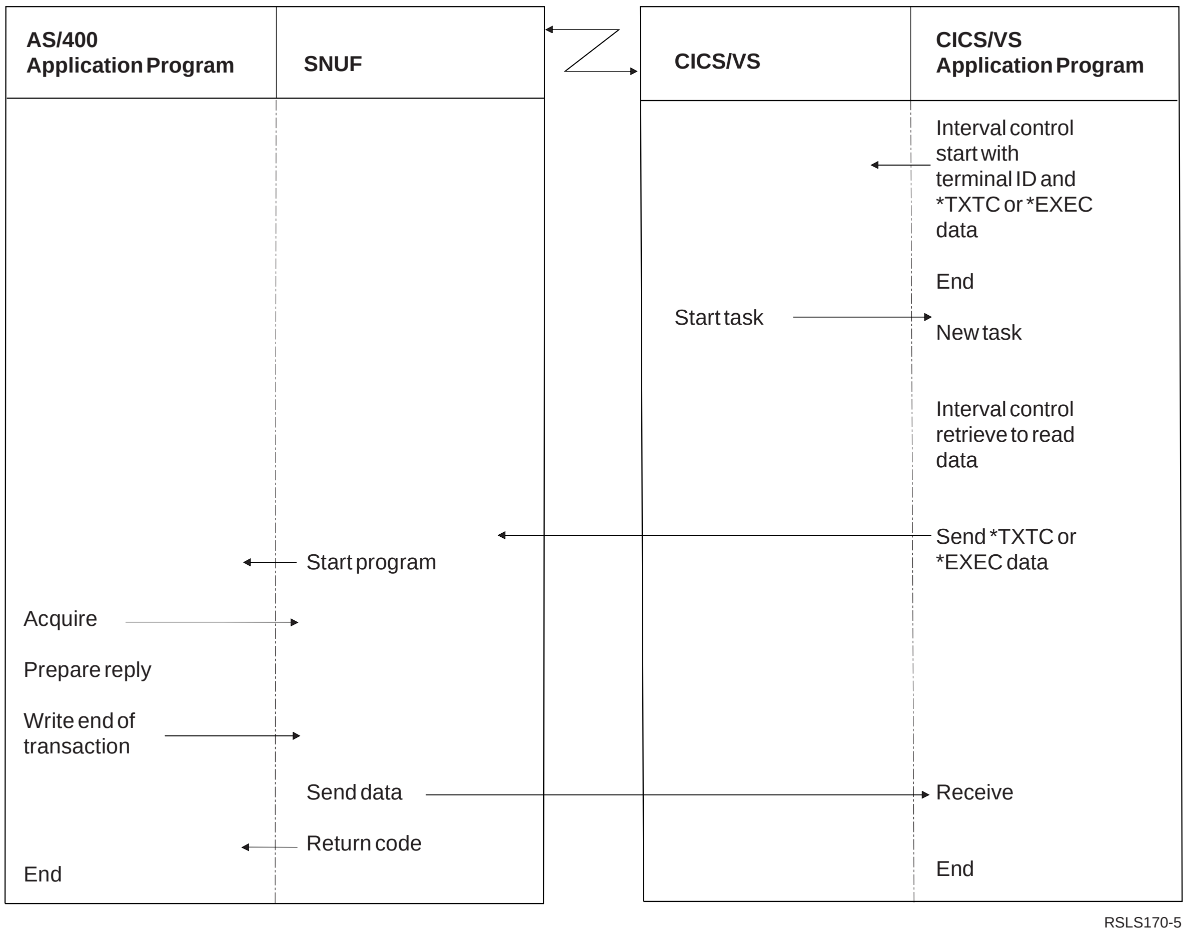


Figure C-6. Program Start Request Using Interval Control Start Command

Programming for IMS/VS Systems

You must define each session during IMS/VS system creation using the TERMINAL and NAME macro instructions.

- The NAME macro defines the logical terminal (LTERM) name of the session.
- The TERMINAL macro defines session parameters to IMS/VS. Each session is defined as an SLUTYPEP terminal.

The following parameters apply to SNUF sessions.

NAME parameter
This parameter specifies the VTAM node name from VTAM/NCP creation.

MSGDEL parameter
This parameter specifies which message types IMS/VS should discard for this logical terminal:

SYSINFO: Specify this attribute to delete all system messages before they are sent to your program. This attribute is recommended for the program start logical units.

NONIOPCB: Specify this attribute to delete the following messages before they are sent to your programs:

- Message switches
- Messages inserted by an IMS/VS program to an alternative PCB
- /BROADCAST messages
- DFS059 TERMINAL status messages

This attribute is recommended for all systems except program start logical units.

NOTERM: Specify this attribute to send only the following messages to your program:

- Message switches
- Messages inserted by an IMS/VS program to an alternative PCB
- /BROADCAST messages

- DFS059 TERMINAL status messages

NONE: Specify this attribute to send all messages to your program.

COMPT1, COMPT2, COMPT3, and COMPT4 parameters
These parameters define up to four separate components for each session. SNUF provides no explicit support for multiple components per session. Therefore, define only one component per session. Additional options include component protection and blocking. SNUF does not provide explicit support for distributed presentation or SNA character string processing. Use these options only if the AS/400 program can handle them.

OPTIONS parameter
This parameter specifies the following additional parameters:

Response Mode parameter
This parameter specifies the response mode for this session (see “Operating in Terminal Response Mode” on page 5-11 for more details).

MFS/NOMFS parameter
This parameter specifies if message format services are provided for this session. If you specify message format services, the AS/400 application must be prepared to handle the data streams.

ACK/OPTACK parameter
This parameter specifies the type of response required. You must select OPTACK for SNUF. If you select ACK, the session ends when an update or recoverable transaction is attempted.

BID/NOBID parameter
This parameter specifies if the VTAM BID command is used.

OPNDST/NOPNDST parameter
This parameter specifies if sessions can be started with the /OPNDST command. Select OPNDST for program start sessions because they may have to be started from the host system.

OUTBUF parameter
This parameter specifies the maximum request unit size for the BIND command. SNUF rejects any BIND command with a maximum request unit greater than 4096 bytes (the default value is 256).

Figure C-7 is a portion of an IMS/VS definition. The highlighted parameters correspond to parameters on the OVRICFDEVE and ADDICFDEVE commands and are described after the figure.

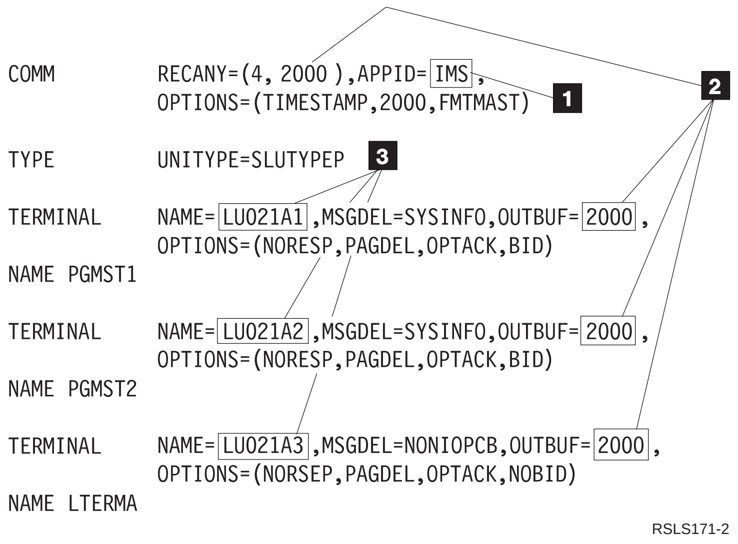

```
COMM        RECANY=(4, 2000 ),APPID=IMS ,
            OPTIONS=(TIMESTAMP,2000,FMTMAST)

TYPE        UNITYPE=SLUTYPEP

TERMINAL    NAME=LU021A1,MSGDEL=SYSINFO,OUTBUF=2000,
            OPTIONS=(NORESP,PAGDEL,OPTACK,BID)
NAME PGMST1

TERMINAL    NAME=LU021A2,MSGDEL=SYSINFO,OUTBUF=2000,
            OPTIONS=(NORESP,PAGDEL,OPTACK,BID)
NAME PGMST2

TERMINAL    NAME=LU021A3,MSGDEL=NONIOPCB,OUTBUF=2000,
            OPTIONS=(NORSEP,PAGDEL,OPTACK,NOBID)
NAME LTERMA
```

Figure C-7. Example of IMS/VS Definition Parameters

In Figure C-7 on page C-9:

1 This name corresponds to APPID on the CRTDEVSNUF and OVRICFDEVE commands.

2 For best performance, the buffer size should be large enough to handle the maximum user record length sent by the AS/400 system, as defined by RCDLEN on the CRTDEVSNUF or OVRICFDEVE commands.

3 In this example, local addresses 1 and 2 (LU021A1 and LU021A2) are used for program start sessions. Local address 3 (LU021A3) is used for acquired sessions.

Program Start Request Considerations

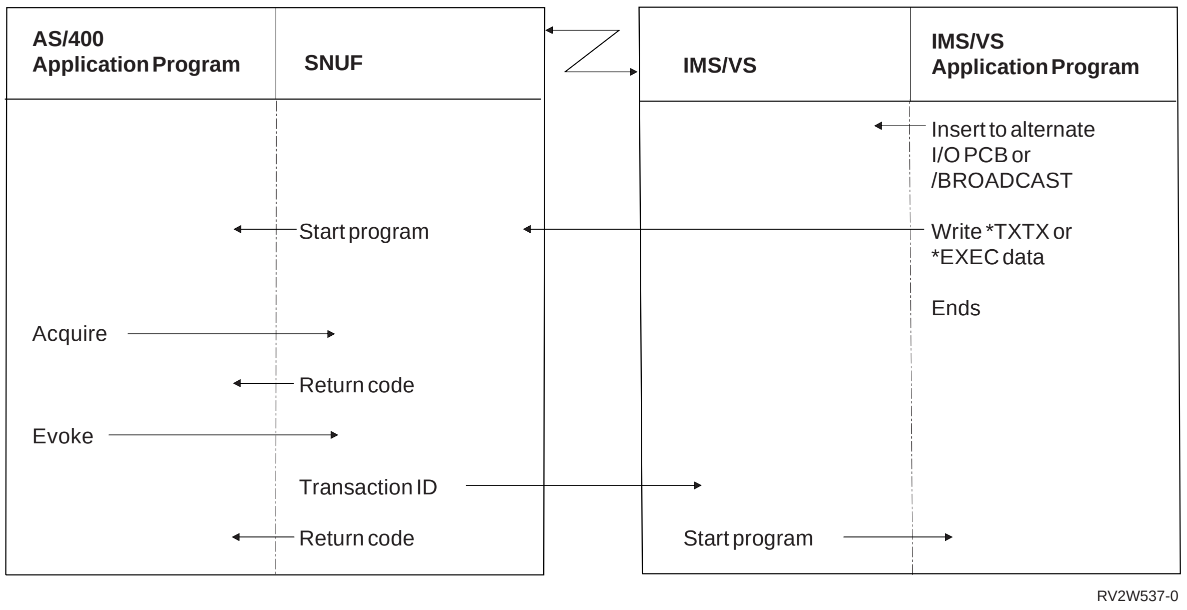
SNUF accepts program start requests only on sessions reserved for this purpose. Logical units that send program start requests to the AS/400 system must use the MSGDEL=SYSINFO option on the TERMINAL macro.

The logical units reserved for program start requests must be started from the host system by the VARY command, the LOGAPPL parameter in the VTAM definition, or the IMS/VS /OPNDST command.

IMS/VS programs that use the program start request must have defined a modifiable alternative program control block in the program specification block. For example, for the IMS/VS program to start a program on an AS/400 system using LTERM=S3XA, it must first perform a Change Call (CHNG) operation to set the destination of the alternative program control block to S3XA. The program then performs an Insert Call (ISRT) operation to include the name of the process to be started and any security information or parameters required. The program can then perform Insert Call operations to build an output message.

The IMS/VS program cannot receive data through the alternative program control block. Therefore, the program started on an AS/400 system cannot reply to the message it received from IMS/VS through the same logical unit from which it was received. However, the program can acquire another session and start a transaction on that session to send a reply.

Example IMS/VS Remote Program Start Request: Figure C-8 is an example of how an IMS/VS program starts an AS/400 program and how the AS/400 program attaches to the IMS/VS application program.



AS/400 Application Program
SNUF
IMS/VS
IMS/VS Application Program
Insert to alternate I/O PCB or /BROADCAST
Start program
Write *TXTX or *EXEC data
Ends
Acquire
Return code
Evoke
Transaction ID
Return code
Start program
RV2W537-0

Figure C-8. Example IMS/VS Remote Program Start Request

Appendix D. SNA 3270 Program Interface

The **SNA 3270 program interface** for SNA upline facility (SNUF) allows an AS/400 application to communicate with a host application by sending and receiving 3270 data streams. For additional information on the 3270 work station, 3270 devices and 3270 data streams, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide* and the *IBM 3270 Information Display System, Data Stream Programmer's Reference*.

Notes:

1. SNUF is not intended to emulate a 3174 or 3274 Control Unit.
2. If you are using 3270 emulation for BSC, see the *IBM System/38 Data Communications Programmer's Guide* and the *IBM System/38 3270 Emulation Reference Manual and User's Guide*.

The application programmer constructs the AS/400 system application using AS/400 system support of programs, intersystem communications function (ICF) files and program devices. The programs can be coded in any high-level language that supports the ICF. The application program may be interactive, or it may be started from the host system through a program start request.

A typical AS/400 system interactive application might consist of high-level language programs (possibly with supporting CL programs) that access database, display, and print files. To communicate with a host system through SNA 3270 program interface, the application program must use an ICF file.

A typical application might open the ICF file, acquire the program device session, send and receive data from the host application, send or receive a DETACH to end the transaction with the host application, and release the session with the host system.

For each ICF operation the application program should monitor for a successful major or minor return code. The data flow is indicated in a required 20- or 32-byte common header for both sent and received data. The SNA 3270 program interface handles data as a 3274 Type C (remote) unit.

Note: The 3274 models bearing the letter designation "C" are: 1C, 21C, 31C, 41C, 51C, and 61C. These Type C units operate as remote units using either synchronous data link control (SDLC) or binary synchronous communications (BSC). The application program sends and receives data using Write, Erase/Write, Erase/Write Alternate, Erase All Unprotected, Read Modified, Read Modified All, Read Buffer, and Write Structured Field 3270 data stream commands.

ICF File Considerations

The EMLDEV parameter on the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands indicates if the device is used to send and receive 3270 data streams and, if so, whether and how the data stream is formatted.

The EMLDEV parameter specifies both the emulated device type (with a default of no 3270 emulation) and 3270 data streams for the following supported printers and displays:

- 3278, Models 2, 3, 4, and 5
- 3284
- 3286
- 3287
- 3288
- 3289

The second part of the EMLDEV parameter, the data format, specifies the format of the 3270 data stream being sent and received by the application program. In particular, it specifies the form the 3270 data stream takes in the application I/O buffer. The data stream interface with the user program can either be formatted or unformatted.

The unformatted interface presents the 3270 data stream to your program in its raw form exactly as it is sent from the host system. To use the raw data, you should know the following:

- The format of the 3270 data stream orders
- The function of the 3270 data stream orders
- The 3270 addressing scheme
- The 3270 attribute bit assignments

Using the formatted interface reduces the need to know the 3270 data stream orders, format, function, and so on.

The formatted interface translates the 3270 data stream and builds a 1920, 2560, 3440, or 3564 character image of the display, which is then presented to your application program. The formatted interface allows you to request 3270 display images with or without field definitions. Field definitions are built following the display image. For display emulation, each field definition identifies the location and characteristics of a particular field in the display image. For printer emulation, a field definition is also used to identify a printer order.

EMLDEV

The presence of this parameter with values other than *NONE indicates that the SNA 3270 program interface is using this program device. The EMLDEV parameter is available with the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands.

For the SNA 3270 program interface, the parameter may have two parts. The first part indicates the intended device for which 3270 data streams will be transmitted between an AS/400 system user application program and a host application program.

***NONE**: This default value indicates that the program device entry is not used to send and receive 3270 data streams.

***3278** Specifies the data stream is for a 3277, 3278, or 3279 Display Station.

***3284** Specifies the data stream is for a 3284 Printer Device.

***3286** Specifies the data stream is for a 3286 Printer Device.

***3287** Specifies the data stream is for a 3287 Printer Device.

***3288** Specifies the data stream is for a 3288 Printer Device.

***3289** Specifies the data stream is for a 3289 Printer Device.

Note: If a value other than *NONE is specified, the second part of the parameter must be specified.

The second part of the parameter value indicates whether the 3270 data streams transmitted are formatted or unformatted and if field definitions are included when using formatted data. The following values are accepted for the second part of the parameter value.

***UNFORMAT**: This default value specifies that the application programs send and receive data in unformatted form with control information embedded.

***FIELD**: Specifies that the AS/400 system user application program sends and receives a formatted version of the 3270 data streams, which are transmitted between the user program and the host application program. The 3270 data stream will be formatted for the application program in Display/Printer Image form, with control information followed by field definitions of 8 bytes each that indicate the location and characteristics of fields or printer orders.

***NOFIELD**: Specifies that the AS/400 system user application program sends and receives a formatted version of the 3270 data streams, which are transmitted between the user program and a host application program. The 3270 data stream will be formatted for the application program in Display/Printer Image form, with control information but without field definitions that indicate the location and characteristics of fields or printer orders.

***EXTFIELD**: Specifies that the AS/400 system user application program sends and receives a formatted version of the 3270 data streams, which are transmitted between the user program and a host application program. The 3270 data stream will be formatted for the application program in 3278 display image form, with control information followed by extended field definitions of 10 bytes each that indicate the location and characteristics of fields, plus the three extended field attribute bytes.

Note: *EXTFIELD is valid only if *NO is specified on the *BATCH parameter and if *3278 is specified on the EMLDEV parameter.

Writing Application Programs Using SNA 3270 Program Interface

The AS/400 system application programmer must know what is happening in the host application and on the host end of the line. Your application program must check for, and respond to, ICF major/minor return codes resulting from input/output operations with the host program, and it may monitor for messages. Your program may access the data flow, SSCP-LU or LU-LU, indicated in the 20-or 32-byte common header in the buffer. See “3270 Data Flow” on page D-14 for a complete list of indicators. (In addition to the normal data LU-to-LU and SSCP-to-LU flows, there is a special data flow indication to inform the program when a query response is received as a reply to a get operation during the LU-to-LU flow.)

As an aid for new applications, it is suggested you run a communications line trace of the host application being run to and from the actual 3270 device. This can be compared against the SNA 3270 program interface running the same host program. This should aid in later debugging of the new SNA 3270 program.

When using the unformatted interface emulating a display rather than a printer EMLDEV(*3278 *UNFORMAT), the AS/400 system application program must be designed to provide a 20-byte common header and a 3270 data stream in the format the host system is expecting to receive. The 3270 data stream will consist of a write of user data and necessary control characters associated with a 3278 display device.

When using the unformatted interface emulating either a display or a printer, your program reads and extracts user data from the 3270 display or printer data stream.

When using the formatted interface EMLDEV(*32xx *NOFIELD), EMLDEV(*32xx *FIELD), or EMLDEV(*3278 *EXTFIELD), your program reads the data in the common header, display or printer image, and field definitions on a formatted read. It changes the necessary fields before returning the same structures and data on a formatted write to the host system.

The normal display or printer image you receive when working with the formatted interface is a 1920-byte image. For non-3278 displays this value is always 1920. It is possible to receive larger images from the host application program. For 3278 displays the value may be 1920, 2560, 3440, or 3564 bytes. This is specified on byte 24 on the BIND command received from the host system. See Figure D-12 on page D-14 for more information on the BIND command received from the host system. Also, any display image received on the SSCP-LU flow is not more than 1920 bytes. This applies to images received before or after the BIND command. The block size in the formatted header informs you of the image size.

When your program acquires a program device specified for SNA 3270 program interface (Display) and sign-on text is sent to the host system with an application identifier (other than *USER) for a host application program, SNUF expects a BIND command from the remote system. When you specify the application identifier parameter as *USER on the ICF device entry, SNUF does not send a logon command to the host system. Instead, the SNA 3270 program interface receives and handles SSCP-LU USSMSG messages including sending a negative response, if appropriate. If your application program sends a logon command to the host system after receiving the USSMSG message, SNUF sends a positive response to the USSMSG message and then issues the logon command to the host system. An application identifier of *USER is only meaningful if you are using the 3270 program interface.

All USSMSG messages sent by the host to a SNA Upline Facility 3270API program, with the application identifier option set to *USER, must be received by the AS/400. This is done by using a READ operation instead of a READ FROM INVITED PROGRAM DEVICE operation.

If the 3270API application sends the host a logon message, the message was sent using a PUT operation instead of a PUT-INVITE operation.

When your program acquires a program device specified for SNA 3270 program interface (Printer), SNUF expects a BIND command without sending sign-on text.

In general, an AS/400 system application program must be designed to handle two types of data streams sent from the host system:

- A data stream from a host write operation
- A data stream that contains a host read

For either, the user program must respond by doing a Read command to receive the data, the 3270 command, or both.

Unformatted Program Interface

When EMLDEV(*32xx *UNFORMAT) has been specified on the ADDICFDEVE or CHGICFDEVE command, the user program will receive and send a common header followed by the unformatted 3270 data stream. In this case, the only significant fields in the common header are the 3270 command and the data flow indication. The common header for the unformatted program interface is illustrated in Figure D-1.

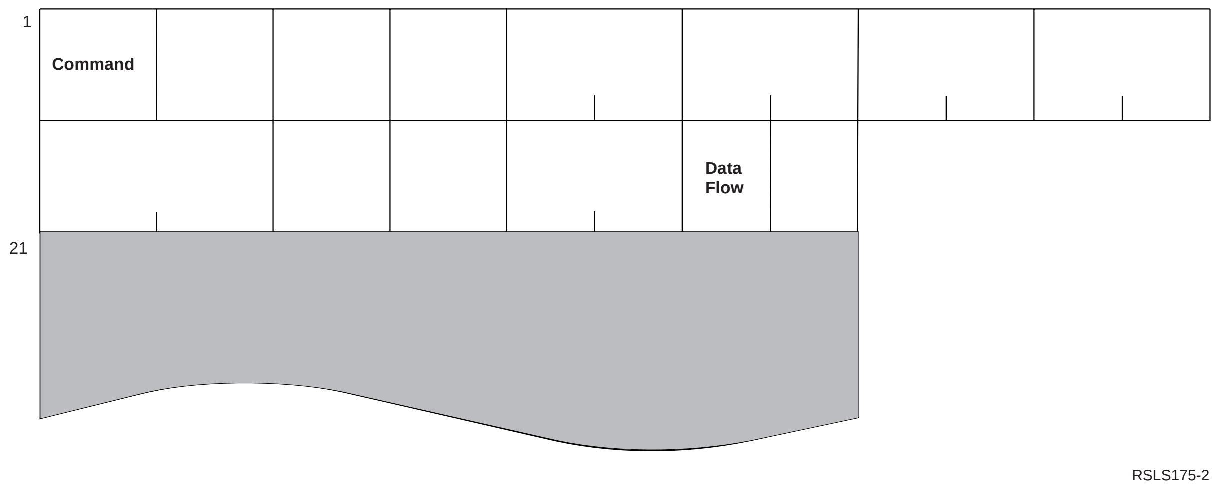


Figure D-1. Common Header for Unformatted Program Interface (20 Bytes)

The common header fields have the following meanings:

- **Byte 1:** The CMD (Command) has the same meaning as in the 3270 data stream. Byte 1 is copied from the 3270 data stream received by the AS/400 system from the host system. On a write to the host system it may be set or changed by the user program.
- **Byte 19:** This field indicates the data flow that is currently active, either LU-LU or SSCP-LU. See "3270 Data Flow" on page D-14.

Figure D-2 is an unformatted 3270 data stream received from the System/370 host system on a host write operation (as it would be received in the user program buffer following the header):

Figure D-2. AS/400 Program Read (To a Write Command)

Data Stream Command/Order	Bytes	Functional Description
Write-Type Command Code	1	Write, Erase/Write, Erase/Write Alternate, Write Structured Field
WCC	1	Write control character
Orders and data	N	Printout format orders or buffer control orders with data

For a description of printer formatting orders (NL, EM, FF, SI, and CR), buffer control orders (SBA, SF, IC, PT, RA, EUA), and orders for Structured Field and Attribute processing (SFE, MF, SA), see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*.

Figure D-3 on page D-5 is a 3270 data stream for a System/370 host read request operation:

Figure D-3. AS/400 Read (To a Read Request Command)

Data Stream Command/Order	Bytes	Functional description
Read-Type Command Code	1	Read Buffer, Read Modified, or Read Modified All

Figure D-4 is a 3270 data stream on a write operation from an AS/400 system program in response to a System/370 host Read Buffer command:

Figure D-4. AS/400 Program Write (To a Read Buffer Command)

Data Stream Command/Order	Bytes	Functional description
AID	1	No AID generated (Display), No AID (Printer), Enter, PF or PA
Cursor Address	2	Cursor Address
SF	1	Start Field, or Start Field Extended
Field Attribute Character	1	Attribute Character
Data	data length	Data
SF	1	Start Field, or Start Field Extended
Field Attribute Character	1	Attribute Character
Data	data length	Data

Figure D-5 is a 3270 data stream on a write operation from an AS/400 system program in response to a System/370 host Read Modified or Read Modified All command. In this data stream, only modified data fields are expected by the host system.

Note: The AID is other than the CURSR SEL key, a PA key or the CLEAR key.

Figure D-5. AS/400 Program Write (To a Read Modified Command)

Data Stream Command/Order	Bytes	Functional description
AID	1	No AID generated (Display), No AID (Printer), Enter, PF key
Cursor Address	2	Cursor Address
SBA	1	Set Buffer Address
Buffer Address	2	Buffer Address
Data	data length	Data
SBA	1	Set Buffer Address
Buffer Address	2	Buffer Address
Data	data length	Data

If the AID is a PA key or the CLEAR key, then the 3270 data stream in Figure D-6 is written from an AS/400 system program in response to a System/370 host Read Modified or Read Modified All command:

Figure D-6. AS/400 Program Write (To a Read Modified Command)

Data Stream Command/Order	Bytes	Functional description
AID	1	PA key or CLEAR key

The AID is for the CURSR SEL key.

Figure D-7. AS/400 Program Write (To a Read Modified Command)

Data Stream Command/Order	Bytes	Functional description
Cursor Select AID	1	No AID generated (Display), No AID (Printer), Enter, PF key
Cursor Address	2	Cursor Address
SBA	1	Set Buffer Address
Buffer Address	2	Buffer Address
SBA	1	Set Buffer Address
Buffer Address	2	Buffer Address

Formatted Program Interface

When EMLDEV (*32xx *NOFIELD), EMLDEV (*32xx *FIELD), or EMLDEV (*3278 *EXTFIELD) is specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, the user program receives and sends 3270 data stream information through the formatted program interface.

EMLDEV (*32xx *NOFIELD): When EMLDEV (*32xx *NOFIELD) is specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, the formatted program interface has the buffer content shown in Figure D-8.

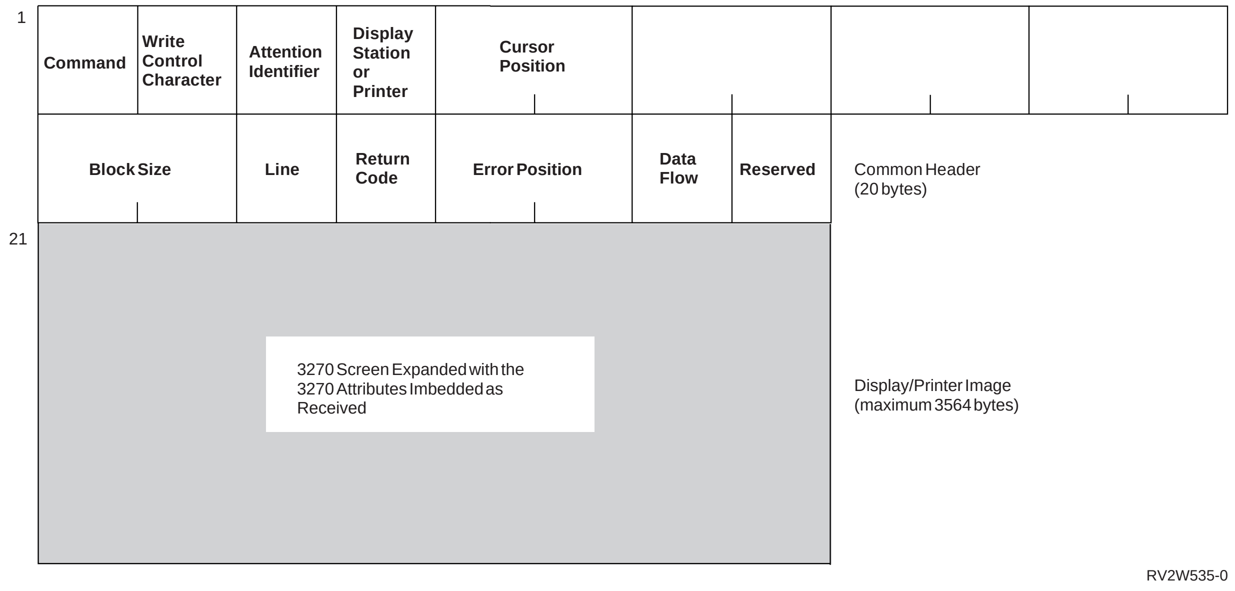


*Figure D-8. Common Header and Buffer for Formatted Program Interface—EMLDEV (*3278 *NOFIELD)*

In the common header, certain fields are set by the AS/400 system when a 3270 data stream is received from the host system, some are set by the user program when sending a 3270 data stream to the host system, and some are set by both.

If the first operation performed by the user program is a write, fields in the header that are set only by the AS/400 system should be binary zero.

The common header fields have the following meanings:

- **Bytes 1–3:** The Command (CMD), Write Control Character (WCC), and Attention Identifier (AID) have the same meaning as in the 3270 data stream. For a description, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*. Bytes 1–3 are copied from the 3270 data stream received by AS/400 system from the host system. On a write to the host system, they may be set or changed by the user program.
- **Byte 4:** Display Station or Printer is the device type. If this field is the character 0, this device is a display. If this field is the character 1, this device is a printer. The AS/400 system sets this field when the first 3270 data stream is received from the host system. The user program should return the same value on subsequent writes to the host system.
- **Bytes 5–6:** Cursor Position is the position (binary 1 through block size) where the cursor is located. This value is set by the AS/400 system when a 3270 data stream is received from the host system (if an insert cursor order, IC, is not found in the data stream, the cursor remains at its last location) and set or changed on a user program write to the host system.
- **Bytes 7–12:** These are set only when the user has specified EMLDEV(*32xx *FIELD or *EXTFIELD) on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

- **Bytes 13–14:** Block Size is a binary number indicating the size (in bytes) of the data in the display or printer image. This value, added to the header length, indicates the last position in the input buffer into which data from the host system is placed.

  If the block size is less than 1920, all bytes in the display image buffer, after the block size, contain nulls. For displays, the value size may increase from 1920 bytes to 2560, 3440, or 3564 bytes if the alternate screen size is used. For printers, this value can be 0 through 1920.

- **Byte 15:** Line (LIN) is the size of the character print line and only applies to printers. This value indicates how data should be formatted if it is printed. Possible line values are:

  1 for 40 characters per line
  2 for 64 characters per line
  3 for 80 characters per line
  4 for unformatted (Line size is determined by the new line, NL, print order)

- **Byte 16:** Return Code (RET) contains return code information about translation. This field should always be set to binary zero when issuing read or write operations. Possible values are:

'00'X	No condition.
'01'X	Not enough space for all field entries; some created but more are possible. This value applies to Read operations only.
'02'X	Incorrect attention identifier key field; this value applies only to write operations.
'03'X	Incorrect cursor position; this value applies only to write operations.
'04'X	Attributes changed in the display image buffer to a value other than 01; this value applies only to write operations.
'05'X	Number of entries is larger than what was previously returned; this value applies only to write operations.

- **Bytes 17–18:** Error Position (ERR) is a binary number indicating the error position in the display image buffer that corresponds to the return code. For a display device, this position is that of an attribute. For a printer, this position may be either the position of an attribute or the position of a printer order. This field applies only when the value returned in the return code position is 1 or 4.

- **Byte 19**: This field indicates the data flow that is currently active, either LU-LU, QUERY REPLY, or SSCP-LU. See “3270 Data Flow” on page D-14.

- **Byte 20:** This is a reserved space and must contain binary zeros.

Display Image: The display image in Figure D-8 on page D-6 contains 1920, 2560, 3440, or a maximum of 3564 bytes of attributes and display or printer data. Each attribute occupies one location in the buffer. An attribute character defines the beginning of a field and contains characteristics about the field. The attribute character has the same meaning as bits 2 - 7 of the attribute character in the 3270 data stream. For a description of the attribute character bit definitions, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer’s Guide*.

When EMLDEV(*32xx *NOFIELD) is specified, the user program must update the attribute byte and the field itself when it modifies fields to be sent to the host system. Bit 7 of the attribute byte must be set to indicate a modified field.

EMLDEV (*32xx *FIELD): When EMLDEV(*32xx *FIELD) is specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, the formatted program interface has the buffer content shown in Figure D-9 on page D-8.

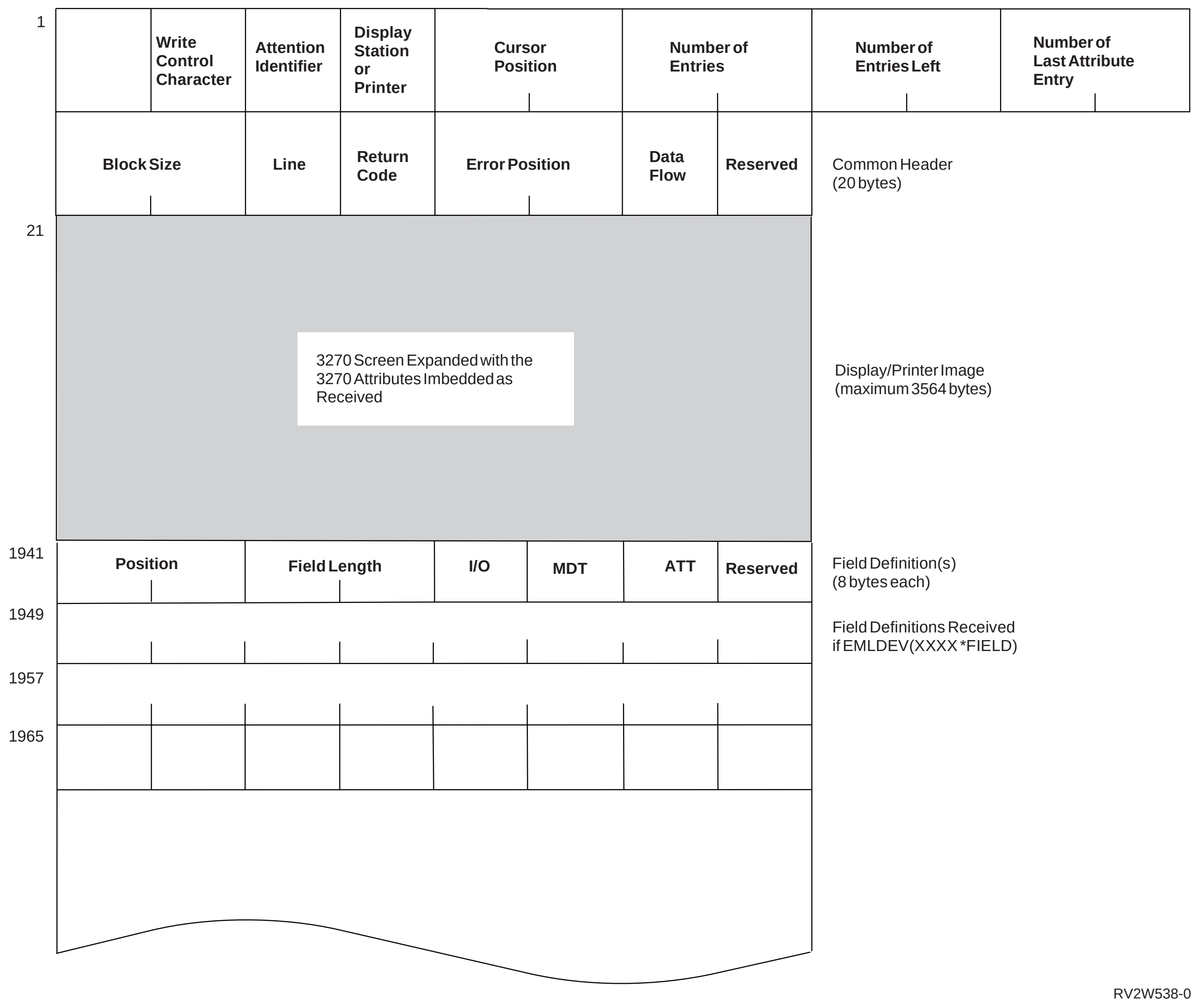

1
Write Control Character
Attention Identifier
Display Station or Printer
Cursor Position
Number of Entries
Number of Entries Left
Number of Last Attribute Entry
Block Size
Line
Return Code
Error Position
Data Flow
Reserved
Common Header (20 bytes)
21
3270 Screen Expanded with the 3270 Attributes Imbedded as Received
Display/Printer Image (maximum 3564 bytes)
1941
Position
Field Length
I/O
MDT
ATT
Reserved
Field Definition(s) (8 bytes each)
1949
Field Definitions Received if EMLDEV(XXXX *FIELD)
1957
1965
RV2W538-0

*Figure D-9. Common Header and Buffer for Formatted Program Interface—EMLDEV (*32xx *FIELD)*

In the common header, certain fields are set by the AS/400 system when a 3270 data stream is received from the host system, some are set by the user program when sending a 3270 data stream to the host system, and some are set by both.

If the first operation performed by the user program is a write, fields in the header that are set only by the AS/400 system should be binary zero.

The common header fields have the following meanings:

- **Bytes 1–3:** The Command (CMD), Write Control Character (WCC), and Attention Identifier (AID) have the same meaning as in the 3270 data stream. For a description, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*. Bytes 1–3 are copied from the 3270 data stream received by AS/400 system from the host system. On a write to the host system, they may be set or changed by the user program.
- **Byte 4:** Display Station or Printer is the device type. If this field is the character 0, this device is a display. If this field is the character 1, this device is a printer. The AS/400 system sets this field when the first 3270 data stream is received from the host system. The user program should return the same value on subsequent writes to the host system.
- **Bytes 5–6:** Cursor Position is the position (binary 1 through block size) where the cursor is located. This value is set by the AS/400 system when a 3270 data stream is received

from the host system (if an insert cursor order, IC, is not found in the data stream, the cursor remains at its last location) and set or changed on a user program write to the host system.

- **Bytes 7–8:** Number of Entries is a binary number indicating the total number of field definitions including attribute entries and printer order entries. When data is received from the host system, the AS/400 system returns the number of entries created. On a write to the host system, the user program sets this value to the number of entries to be processed.

  If this value is zero when transmitting data, the AS/400 system returns only the attention identifier, cursor position, and any host set modified data tag fields to the host system.

  If this field is changed to a value greater than the value returned with a previously received 3270 data stream from the host system, an error return code is created.

- **Bytes 9–10:** Number of Entries Left is a binary number indicating the total number of field definitions that could not be built in the user buffer. This number includes both attribute entries and printer order entries.

  This value is set only when a return code result indicates not enough space in the buffer to build field definitions for all fields and printer orders.

- **Bytes 11–12:** Number of Last Attribute Entry is a binary number indicating the number of the field definition that identifies the last attribute in the display image buffer. If the display is not formatted (contains no attributes), this value is zero.

- **Bytes 13–14:** Block Size is a binary number indicating the size (in bytes) of the data in the display or printer image. This value, added to the header length, indicates the last position in the input buffer into which data from the host system is placed.

  If the block size is less than 1920, all bytes in the display image buffer, after the block size, contain nulls. For displays, the value size may increase from 1920 bytes to 2560, 3440, or 3564 bytes if the alternate screen size is used. For printers, this value can be 0 through 1920.

- **Byte 15:** Line (LIN) is the size of the character print line and only applies to printers. This value indicates how data should be formatted if it is printed. Possible line values are:

  1 for 40 characters per line
  2 for 64 characters per line
  3 for 80 characters per line
  4 for unformatted (Line size is determined by the new line, NL, print order)

- **Byte 16:** Return Code (RET) contains return code information about translation. This field should always be set to binary zero when issuing read or write operations. Possible values are:

'00'X	No condition.
'01'X	Not enough space for all field entries; some created but more are possible. This value applies to Read operations only.
'02'X	Incorrect attention identifier key field; this value applies only to write operations.
'03'X	Incorrect cursor position; this value applies only to write operations.
'04'X	Attributes changed in the display image buffer to a value other than 01; this value applies only to write operations.
'05'X	Number of entries is larger than what was previously returned; this value applies only to write operations.

- **Bytes 17–18:** Error Position (ERR) is a binary number indicating the error position in the display image buffer that corresponds to the return code. For a display device, this position is that of an attribute. For a printer, this position may be either the position of an attribute or the position of a printer order. This field applies only when the value returned in the return code position is 1 or 4.

- **Byte 19**: This field indicates the data flow that is currently active, either LU-LU, QUERY REPLY, or SSCP-LU. See “3270 Data Flow” on page D-14.

- **Byte 20:** This is a reserved space and must contain binary zeros.

Display Image: The display image in Figure D-9 on page D-8 contains 1920, 2560, 3440, or a maximum of 3564 bytes of attributes and display or printer data. Each attribute occupies one location in the buffer. An attribute character defines the beginning of a field and contains characteristics about the field. The attribute character has the same meaning as bits 2–7 of the attribute character in the 3270 data stream. For a description of the attribute character bit definitions, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*.

When EMLDEV(*32xx *FIELD) is specified, the user program must update the attribute byte, the field itself, and the field definition when it modifies fields to be sent to the host system.

The modified data tag byte (MDT) of the field definition entry must be set to '01'X, or an error (RET = 4) will occur.

Field Definitions: The field definitions in Figure D-9 describe the input and output field for display devices and printer orders for printer devices. Field definitions apply when EMLDEV(*32xx *FIELD) has been specified.

A field definition entry is created for each field in the display image and is also created for printer orders in a printer data stream, unless the orders are consecutive. Field definitions are ordered from position 1 through 1920, 2560, 3440, or 3564 of the display image.

Whenever data is received from the host system, fields in the display image may be overwritten. When an attribute for a field is overwritten, the field definition corresponding to that field is removed. If the field definition is for an attribute, its format is as follows:

- **Bytes 1–2:** Position is a binary number that defines the relative position from the beginning of the display image to the first character of the field. The values for position can range from 1 through 1920, 2560, 3440, or 3564.
- **Bytes 3–4:** Field Length is a binary number indicating the length of the field defined by the attribute. This length does not include the attribute byte.
- **Byte 5:** I/O indicates whether the field is an unprotected field or a protected field. The I/O entry is the character 0 if the field is unprotected and the character 1 if it is protected.
- **Byte 6:** MDT (Modified Data Tag) indicates whether the field is marked as changed. MDT is '01'X if the field is changed or '00'X if the field is unchanged.

  When receiving data, the MDT value is '01'X if the host system turned on the MDT bit in the attribute, or if the MDT was on from a previous translation and the write control character (WCC) does not indicate that the MDT should be reset. Otherwise the MDT value is '00'X.

  When sending data, the value must be changed from '00'X to '01'X if the unprotected field was changed. If the MDT value is changed to a value other than '01'X or is changed to '01'X for a protected field, the change is ignored.
- **Byte 7:** ATT (Attribute) is a copy of the Attribute of this field. For a description of the attribute character bit definitions, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*.
- **Byte 8:** This is a reserved space and must contain binary zeros.

EMLDEV (*3278 *EXTFIELD): When EMLDEV (*3278 *EXTFIELD) is specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, the formatted program interface has the buffer content shown in Figure D-10 on page D-11.

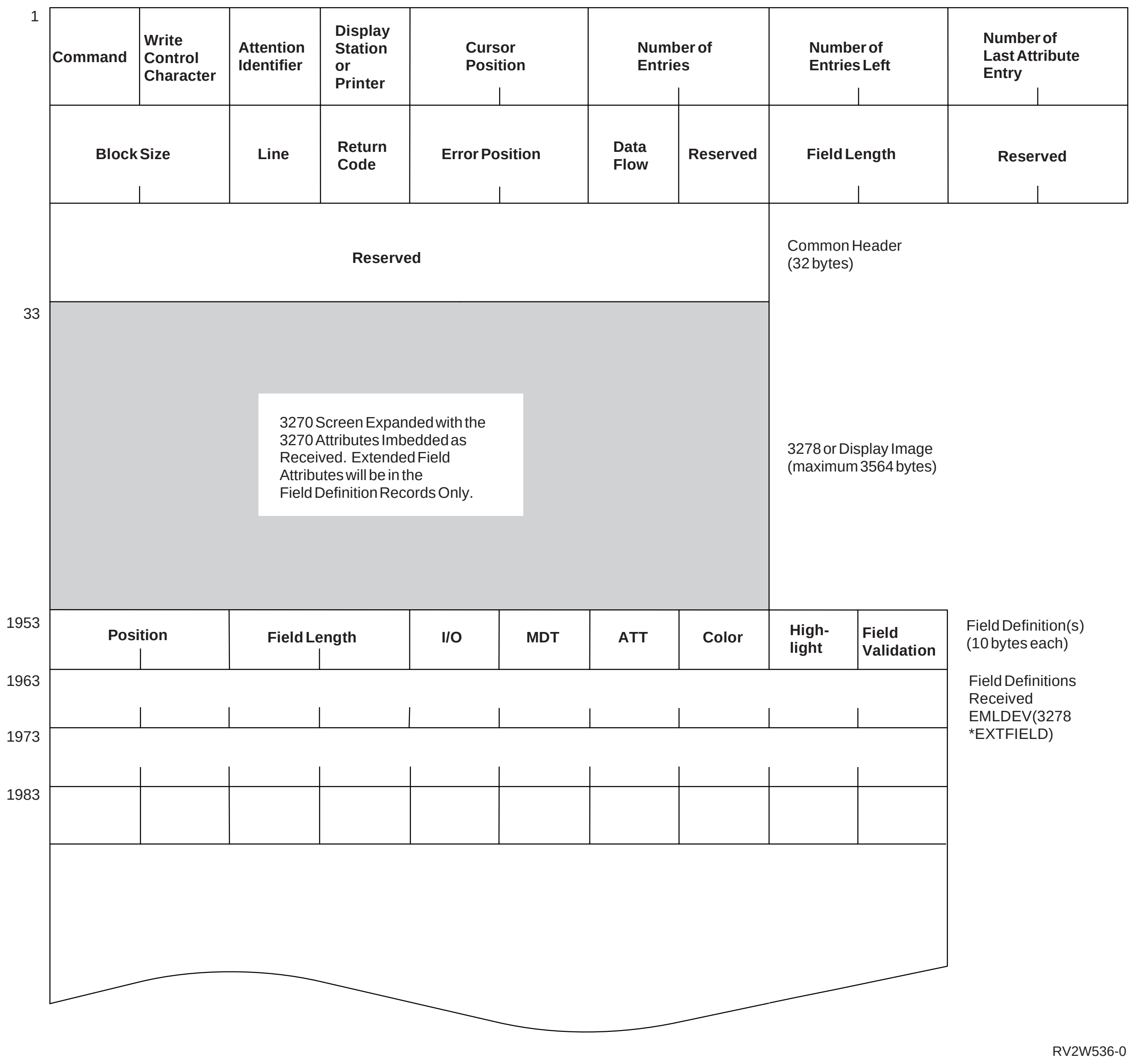

1
Command
Write Control Character
Attention Identifier
Display Station or Printer
Cursor Position
Number of Entries
Number of Entries Left
Number of Last Attribute Entry
Block Size
Line
Return Code
Error Position
Data Flow
Reserved
Field Length
Reserved
Reserved
Common Header (32 bytes)
33
3270 Screen Expanded with the 3270 Attributes Imbedded as Received. Extended Field Attributes will be in the Field Definition Records Only.
3278 or Display Image (maximum 3564 bytes)
1953
Position
Field Length
I/O
MDT
ATT
Color
High-light
Field Validation
Field Definition(s) (10 bytes each)
1963
Field Definitions Received EMLDEV(3278 *EXTFIELD)
1973
1983
RV2W536-0

*Figure D-10. Common Header and Buffer for Formatted Program Interface—EMLDEV (*3278 *EXTFIELD)*

In the common header, certain fields are set by the AS/400 system when a 3270 data stream is received from the host system, some are set by the user program when sending a 3270 data stream to the host system, and some are set by both.

If the first operation performed by the user program is a write, fields in the header that are set only by the AS/400 system should be binary zero.

The common header fields have the following meanings:

- **Bytes 1–3:** The Command (CMD), Write Control Character (WCC), and Attention Identifier (AID) have the same meaning as in the 3270 data stream. For a description, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*. Bytes 1–3 are copied from the 3270 data stream received by AS/400 system from the host system. On a write to the host system, they may be set or changed by the user program.
- **Byte 4:** Display Station or Printer is the device type. If this field is the character 0, this device is a display. If this field is the character 1, this device is a printer. The AS/400 system sets this field when the first

3270 data stream is received from the host system. The user program should return the same value on subsequent writes to the host system.

- **Bytes 5–6:** Cursor Position is the position (binary 1 through block size) where the cursor is located. This value is set by the AS/400 system when a 3270 data stream is received from the host system (if an insert cursor order, IC, is not found in the data stream, the cursor remains at its last location) and set or changed on a user program write to the host system.

- **Bytes 7–8:** Number of Entries is a binary number indicating the total number of field definitions including attribute entries and printer order entries. When data is received from the host system, the AS/400 system returns the number of entries created. On a write to the host system, the user program sets this value to the number of entries to be processed.

  If this value is zero when transmitting data, the AS/400 system returns only the attention identifier, cursor position, and any host set modified data tag fields to the host system.

  If this field is changed to a value greater than the value returned with a previously received 3270 data stream from the host system, an error return code is created.

- **Bytes 9–10:** Number of Entries Left is a binary number indicating the total number of field definitions that could not be built in the user buffer. This number includes both attribute entries and printer order entries.

  This value is set only when a return code result indicates not enough space in the buffer to build field definitions for all fields and printer orders.

- **Bytes 11–12:** Number of Last Attribute Entry is a binary number indicating the number of the field definition that identifies the last attribute in the display image buffer. If the display is not formatted (contains no attributes), this value is zero.

- **Bytes 13–14:** Block Size is a binary number indicating the size (in bytes) of the data in the display or printer image. This value, added to the header length, indicates the last position in the input buffer into which data from the host system is placed.

  If the block size is less than 1920, all bytes in the display image buffer, after the block size, contain nulls. For displays, the value size may increase from 1920 bytes to 2560, 3440, or 3564 bytes if the alternate screen size is used. For printers, this value can be 0 through 1920.

- **Byte 15:** Line (LIN) is the size of the character print line and only applies to printers. This value indicates how data should be formatted if it is printed. Possible line values are:

  1 for 40 characters per line
  2 for 64 characters per line
  3 for 80 characters per line
  4 for unformatted (Line size is determined by the new line (NL) print order)

- **Byte 16:** Return Code (RET) contains return code information about translation. This field should always be set to binary zero when issuing read or write operations. Possible values are:

'00'X	No condition.
'01'X	Not enough space for all field entries; some created but more are possible. This value applies to Read operations only.
'02'X	Incorrect attention identifier key field; this value applies only to write operations.
'03'X	Incorrect cursor position; this value applies only to write operations.
'04'X	Attributes changed in the display image buffer to a value other than 01; this value applies only to write operations.
'05'X	Number of entries is larger than what was previously returned; this value applies only to write operations.

- **Bytes 17–18:** Error Position (ERR) is a binary number indicating the error position in the display image buffer that corresponds to the return code. For a display device, this position is that of an attribute. For a printer, this position may be either the position of an attribute or the position of a printer order.

This field applies only when the value returned in the return code position is 1 or 4.

- **Byte 19**: This field indicates the data flow that is currently active, either LU-LU, QUERY REPLY, or SSCP-LU. See “3270 Data Flow” on page D-14.
- **Byte 20:** This is a reserved space and must contain binary zeros.
- **Byte 21–22:** Field Length is a binary number indicating the length of each field definition that follows the image.
- **Byte 23–32:** This is a reserved space and must contain binary zeros.

Display Image: The display image in Figure D-10 on page D-11 contains 1920, 2560, 3440, or a maximum of 3564 bytes of attributes and display data. Each attribute occupies one location in the buffer. An attribute character defines the beginning of a field and contains characteristics about the field. The attribute character has the same meaning as bits 2–7 of the attribute character in the 3270 data stream. For a description of the attribute character bit definitions, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*.

When EMLDEV(*3278 *EXTFIELD) is specified, the user program must update the attribute byte, the field itself, and the field definition when it modifies fields to be sent to the host system.

The modified data tag byte (MDT) of the field definition entry must be set to '01'X, or an error (RET = 4) will occur.

Field Definitions: The field definitions in Figure D-10 describe the input and output field for display devices and printer orders for printer devices. Field definitions apply when EMLDEV(*3278 *EXTFIELD) has been specified.

A field definition entry is created for each field in the display image. Field definitions are ordered from position 1 through 1920, 2560, 3440, or 3564 of the display image.

Whenever data is received from the host system, fields in the display image may be overwritten. When an attribute for a field is overwritten, the field definition corresponding to that field is removed. If the field definition is for an attribute, its format is as follows:

- **Bytes 1–2:** Position is a binary number that defines the relative position from the beginning of the display image to the first character of the field. The values for position can range from 1 through 1920, 2560, 3440, or 3564.
- **Bytes 3–4:** Field Length is a binary number indicating the length of the field defined by the attribute. This length does not include the attribute byte.
- **Byte 5:** I/O indicates whether the field is an unprotected field or a protected field. The I/O entry is the character 0 if the field is unprotected and the character 1 if it is protected.
- **Byte 6:** MDT (Modified Data Tag) indicates whether the field is marked as changed. MDT is '01'X if the field is changed or '00'X if the field is unchanged.

  When receiving data, the MDT value is '01'X if the host system turned on the MDT bit in the attribute, or if the MDT was on from a previous translation and the write control character (WCC) does not indicate that the MDT should be reset. Otherwise the MDT value is '00'X.

  When sending data, the value must be changed from '00'X to '01'X if the unprotected field was changed. If the MDT value is changed to a value other than '01'X or is changed to '01'X for a protected field, the change is ignored.
- **Byte 7:** ATT (Attribute) is a copy of the Attribute of this field. For a description of the attribute character bit definitions, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*.
- **Byte 8:** Extended color (blue, red, pink, green, turquoise, yellow, and white); attribute type '42'X. For a description of the extended attributes, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*.
- **Byte 9:** Extended highlighting (blink, reverse video, and underline); attribute type '41'X. For a description of the extended attributes, see the *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*.

- **Byte 10:** Field validation (mandatory fill, mandatory entry, and trigger) attribute defines the validation properties of the field in the display image. For a description of the field validation attribute see the *IBM 3270 Information Display System, Data Stream Programmer's Reference*.

3270 Data Flow

Byte 19 of the common header contains a data flow indication. The hexadecimal codes and their descriptions are:

'F0'X	LU-LU. Data flow.
'F1'X	LU-LU. Query reply received. Data flow.
'F9'X	SSCP-LU. Data flow.

The formatted program interface allows an additional indicator, which is the query reply received ('F1'X) indicator. When a reply is received by the application program, it must be sent to the host system with the indicator set to 'F1'X. The reply is provided by the 3270 translation function in response to a query received from the host system.

Host System Programming Considerations

There are considerations for the SNA BIND command, depending on the emulated device entered for the EMLDEV parameter.

Three types of LU-LU sessions are supported. These are:

Type 1	The device is a printer and the data stream is the SNA character string (SCS).
Type 2	The device is a keyboard/display and the data stream is in the 3270 data stream compatibility (DSC) mode format.
Type 3	The device is a printer and the data stream is in the 3270 DSC mode format.

Depending on the device and formatted or unformatted data specified for the EMLDEV parameter, the SNA BIND command is verified for these values listed in Figure D-11.

Figure D-11. BIND Command Byte 14 Values

Device	Format	Byte 14 LU-LU Session Type
3278	*FIELD or *NOFIELD or *UNFORMAT or *EXTFIELD	'02'X
3284	*FIELD or *NOFIELD or *UNFORMAT	'03'X
3286	*FIELD or *NOFIELD or *UNFORMAT	'03'X
3288	*FIELD or *NOFIELD or *UNFORMAT	'03'X
3287	*FIELD or *NOFIELD or *UNFORMAT	'03'X
3287	*UNFORMAT	'01'X
3289	*FIELD or *NOFIELD or *UNFORMAT	'03'X
3289	*UNFORMAT	'01'X

Also, when you specify a keyboard and display (*3278) with formatted data for the EMLDEV parameter, the SNA BIND command specifies the screen size for the display. The value in byte 24 determines if a default size is used or if a value in bytes 20 through 24 is checked for screen size. The sizes in Figure D-12 are supported when a 3278 device is specified. For more information on the BIND command, see the book, *3270 Device Emulation Support*.

Figure D-12. BIND Command Screen Definition

Byte 24 (*3278 Only)	Bytes 20–21	Bytes 22–23
'7E'X	'18'X '50'X [1]	optional value, not checked
'7F'X	optional value, not checked	'18'X '50'X [1]
'02'X	base model 2 default	
'00'X	base default	

Note:

[1] Other possible values are: '20'X '50'X, '2B'X '50'X, '1B'X '84'X

General Considerations

The following information for SNA 3270 support includes considerations for SNUF devices, System/36 restrictions, application identifiers, language operations, keywords, and system-supplied formats.

SNUF Devices

If SNUF device descriptions are to be used for the SNA 3270 program interface, they must be created on a Version 1 Release 2 Modification 0 system or later. If you attempt to use a device created on an earlier release, an 82B5 return code is sent to your program.

System/36 Restrictions

Application programs running in the System/36 environment cannot acquire a SNUF device for the purpose of running a SNA 3270 application. If a System/36 program attempts to open an ICF file to a SNA 3270 program interface device, an 82B4 return code results.

Application Identifiers

If the host application program is the Federal Reserve communications application program, Federal Link Access for Secondary Half-sessions (FLASH), the application identifier (APPID) parameter is checked when the BIND command is received from the host system. The SNUF application program verifies that the BIND command was sent from the correct application at the host site. This verification is done by checking the APPID against the user data field in the BIND command. If the user data field does not exist or does not match the APPID, SNUF compares the APPID against the primary LU name in the BIND command. An INIT-SELF command is created using the APPID value as the primary LU name.

Language Operations, Keywords, and System-Supplied Formats

All data management operations supported by SNUF can be used with SNA 3270 program interface. The SNUF session runs in half-duplex flip-flop protocol. The operations listed in Figure A-1 on page A-1 are supported.

An application program that uses SNA 3270 program interface to exchange 3270 data stream data with a host system program can be coded in any high-level language that can communicate with the host system through SNUF. See Appendix A for supported language operations, DDS keywords, and system-supplied formats.

Example Program

The following example ILE COBOL/400 program communicates with a CICS test program named ID01 which runs on a host CICS system.

1 An ICF file named ICF01 was created for this example. This example assumes that the user has created the SRCFILE and SRCMBR attributes. The program device entry created for this example is named CICSDEV. **2**

3 The ACQUIRE performed for this device entry causes a LOGON APPLID(CICS) to be sent to the host system. Here SNUF uses the APPID parameter on the device description to build the logon command. This device entry is also used for any read. **4**

5 When this entry was created using the ADDICFDEVE command, it specified EMLDEV(*3278 *UNFORMAT). The *UNFORMAT parameter value indicates that data streams will be received in an untranslated format and our program will be responsible for interpreting the 3270 data stream. The 20-byte header which will be received with the data when we read from the host program will provide two fields for us.

6 Byte 1 will provide the command which has been extracted from the 3270 data stream and copied into the header for us.

7 Byte 19 provides us with the data flow. In this example we only expect a HEX F0 which tells us this is an LU-to-LU flow. Note the program does not allow for other flows that might be received.

8 Since BATCH(*NO) was the default, only one Read is necessary to receive the 3270 data stream from the host system which is program ID01's main menu. If *YES had been specified multiple reads might have been necessary depending on the record length determined by the host system. The type of 3278 is one of the valid 3270 display types which may be used. **9**

10 In program APICOB1, note that INPUT-BUFFER redefines OUTPUT-BUFFER and that both records allow for a 20-byte header to precede the 1920 byte screen buffer which will be received and could be sent back to the host system. In APICOB1 we allow for all the header fields that are supported although we only expect two of them to be filled in for us when running in *UNFORMAT mode and a read is performed.

11 OUTPUT-LENGTH must be provided and filled in when writing back to the host system. This length value tells the ICF file how long the combined header and buffer data will be. This length is necessary when using system-supplied formats such as $$SEND.

OUTPUT-LENGTH for *UNFORMAT mode requires a minimum of 20 bytes. OUTPUT-LENGTH for other formats (*FIELD, *NOFIELD, *EXTFIELD) requires a minimum output length equal to the sum of the header plus the screen size.

APICOB1 will open files ICF01 and SHOW, acquire the program device CICSDEV, perform a main routine, close the files and exit the program. **12**

13 The MAIN-ROUTINE handles the receiving of the 3270 data stream and sending the clear key and F-11 key. The routine DSP-IN-DATA is used when receiving data from the host program ID01 to display the first 70 bytes of the data for verification purposes. **14**

15 When sending to CICS or responding to the program ID01 note that the CLR-KEY, ENTER-KEY, and F11-KEY hex value is placed in the AID-BYTE field of DATA-3270-STRCT. Cursor position, if required, is placed in CURPOS field which follows the AID-BYTE. Compare this to the *FORMATTED mode where these values would be placed in the header.

16 When this is done, the value HEX 00 is placed in the O-Command or byte 1 of the header. This is done to be consistent with the handling of the header when running in *FIELD or *NOFIELD formatted mode.

17 The value for the key entered and the program request value ID01 is then placed in the buffer following the header and the $$SEND performed by the SEND-3270 routine. The length value which is set when building the output data is set to include the header length plus the length of the data in the DATA-3270-STRCT that is being sent to the host program.

ICF File Creation

The commands needed to create the descriptions for the SNA 3270 SUPPORT example program are:

```
CRTLINSDLC   LIND(APILIN) RSRCNAME(LIN051) ROLE(*SEC)

CRTCTLHOST   CTLD(APICTLR) LINKTYPE(*SDLC) APPN(*NO)
             LINE(APILIN) STNADR(C1)

CRTDEVSNUF   DEVD(APIDEV) LOCADR(07) RMTLOCNAME(APILOC)
             CTL(APICTLR) PGMSTRRQS(*NO) APPID(CICS)     4
             HOST(*CICS) TEXT('SNUF DEVICE FOR APICOB1')
```

The ICF file ICF01 has been created as a copy of the default ICF file QICDMF. The example program uses system supplied formats, however if we were using a specific format that differed from the system supplied one then a DDS file describing the format would be necessary.

The commands needed to create the program device entry and the ICF file ICF01 would look like this:

```
CRTICFF      FILE(ICF01) SRCFILE(QDDSSRC)  1
             SRCMBR(ICFEXAMP) TEXT('ICF FILE FOR APICOB1 EXAMPLE')

ADDICFDEVE   FILE(ICF01) PGMDEV(CICSDEV) RMTLOCNAME(APILOC)     2
             CMNTYPE(*SNUF) DEV(APIDEV) APPID(*DEVD) HOST(*CICS)
             MSGPTC(*NO) EMLDEV(3278 *UNFORMAT)  5   9
```

The DDS for the display named SHOW which displays input data received from the host program follows:

```
     * *******************************************
     * DSPF NAME      SHOW
     * *******************************************
               R DSP1919                    LOGOUT
                 L1           70   B  4   5
```

Sample Program

```
 IDENTIFICATION DIVISION.
  PROGRAM-ID.                    APICOB1.
*
***************************************************************
* USE SNUF 3270 SUPPORT TO COMMUNICATE WITH CICS USING ICF    *
* PROGRAM DEVICE ENTRY CICSDEV.                               *
*                                                             *
* THE MAIN ROUTINE WILL PERFORM THE FOLLOWING STEPS.          *
* APICOB1 RECEIVES THE WELCOME TO CICS SCREEN AND SENDS A     *
* CLEAR KEY TO THE HOST IN RESPONSE.                          *
* APICOB1 READS IN THE BLANK SCREEN.                          *
* APICOB1 SENDS THE PROGRAM ID ID01 TO CICS. THIS             *
* RESULTS IN THE HOST PROGRAM STARTING AND SENDING A 3270     *
* DATA STREAM TO APICOB1.                                     *
* A READ IS PERFORMED FOR THE MAIN MENU WHICH DISPLAYS THE    *
* PF KEYS AND THEIR FUNCTION.                                 *
* ANOTHER READ IS ISSUED TO RECEIVE THE CHANGE DIRECTION      *
* INDICATOR. AFTER THIS IS RECEIVED, APICOB1 CAN THEN         *
* RESPOND TO THE HOST PROGRAM.                                *
* APICOB1 SENDS A PF-11 KEY TO ID01 WHICH WILL CAUSE          *
* US TO RETURN TO CICS.                                       *
* APICOB1 THEN READS IN THE BLANK SCREEN AND CLOSES FILES.    *
***************************************************************
*
 ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
   SOURCE-COMPUTER.              IBM-AS400.
   OBJECT-COMPUTER.              IBM-AS400.
   SPECIAL-NAMES.                I-O-FEEDBACK  IS IO-FBA
                                 OPEN-FEEDBACK IS OPEN-FBA
                                 REQUESTER IS MY-DISPLAY.
  INPUT-OUTPUT SECTION.
   FILE-CONTROL.
*
****************************************************************
*          F I L E   S P E C I F I C A T I O N S               *
*                                                              *
* ICF01        :  ICF FILE USED TO SEND/RECEIVE DATA           *
* DSPFILE      :  USED TO DISPLAY DATA RECEIVED FROM THE HOST*
****************************************************************
*
     SELECT ICF01 ASSIGN TO WORKSTATION-ICF01-SI
         ORGANIZATION IS TRANSACTION
         CONTROL-AREA IS TR-CTL-AREA
         FILE STATUS IS ICF-STATUS MAJ-MIN.
*
     SELECT SHOW ASSIGN TO WORKSTATION-SHOW
         ORGANIZATION IS TRANSACTION
         CONTROL-AREA IS WS-CTL
         FILE STATUS IS STATUS-DSP.
*
 DATA DIVISION.
*
  FILE SECTION.
*
 FD  ICF01
       LABEL RECORDS ARE STANDARD.
*
 01  ICF-REC.
     05  OUTPUT-BUFFER.
         10  OUTPUT-LENGTH          PIC 9(04).  11
         10  OUTPUT-HEADER.
             15  O-COMMAND          PIC X(01).  16
             15  O-WCC              PIC X(01).
             15  O-AID              PIC X(01).
             15  O-DSP              PIC X(01).
```

```
                15  O-CURPOS.
                    20  O-CURPOS-1      PIC X(01).
                    20  O-CURPOS-2      PIC X(01).
                15  O-NUM-ENT           PIC X(02).
                15  O-NUM-LFT           PIC X(02).
                15  O-LST-ENT           PIC X(02).
                15  O-BLK-SIZ           PIC X(02).
                15  O-LINE              PIC X(01).
                15  O-RTRN-CODE         PIC X(01).
                15  O-ERR-POS           PIC X(02).
                15  O-FLOW              PIC X(01).
                15  FILLER              PIC X(01).
            10  OUTPUT-DATA             PIC X(4072).
        05  INPUT-BUFFER REDEFINES OUTPUT-BUFFER.   10
            10  INPUT-HEADER.
                15  I-COMMAND           PIC X(01).   6
                15  I-WCC               PIC X(01).
                15  I-AID               PIC X(01).
                15  I-DSP               PIC X(01).
                15  I-CURPOS.
                    20  I-CURPOS-1      PIC X(01).
                    20  I-CURPOS-2      PIC X(01).
                15  I-NUM-ENT           PIC X(02).
                15  I-NUM-LFT           PIC X(02).
                15  I-LST-ENT           PIC X(02).
                15  I-BLK-SIZ           PIC X(02).
                15  I-LINE              PIC X(01).
                15  I-RTRN-CODE         PIC X(01).
                15  I-ERR-POS           PIC X(02).
                15  I-FLOW              PIC X(01).   7
                15  FILLER              PIC X(01).
            10  INPUT-DATA.
                15  L01-IN              PIC X(70).
                15  REST-OF-3270        PIC X(1850).
                15  FILLER              PIC X(2156).
*
 FD  SHOW
     LABEL RECORDS ARE OMITTED.
*
 01  DSP-REC.
     05  SHOW-RECORD                PIC X(70).
     05  DSP1919 REDEFINES SHOW-RECORD.
         10 L1-SUB                  PIC X(01) OCCURS 70 TIMES.
*
 WORKING-STORAGE SECTION.
*
 77  ICF-STATUS                      PIC X(02).
 77  CONV-INDX                       PIC 99.
 77  STATUS-DSP                      PIC X(02).
 77  HEX-40                          PIC X VALUE ' '.
 77  HEX-5B                          PIC X VALUE '$'.
 77  HEX-61                          PIC X VALUE '/'.
 77  ENTER-KEY                       PIC X VALUE "'".
 77  F1-KEY                          PIC X VALUE '1'.
 77  F2-KEY                          PIC X VALUE '2'.
 77  F3-KEY                          PIC X VALUE '3'.
 77  F4-KEY                          PIC X VALUE '4'.
 77  F5-KEY                          PIC X VALUE '5'.
 77  F6-KEY                          PIC X VALUE '6'.
 77  F7-KEY                          PIC X VALUE '7'.
 77  F8-KEY                          PIC X VALUE '8'.
 77  F9-KEY                          PIC X VALUE '9'.
 77  F10-KEY                         PIC X VALUE ':'.
 77  F11-KEY                         PIC X VALUE '#'.
 77  F12-KEY                         PIC X VALUE '@'.
 77  F13-KEY                         PIC X VALUE 'A'.
 77  F14-KEY                         PIC X VALUE 'B'.
 77  F15-KEY                         PIC X VALUE 'C'.
 77  F16-KEY                         PIC X VALUE 'D'.
 77  F17-KEY                         PIC X VALUE 'E'.
 77  F18-KEY                         PIC X VALUE 'F'.
 77  F19-KEY                         PIC X VALUE 'G'.
 77  F20-KEY                        PIC X VALUE 'H'.
 77  F21-KEY                        PIC X VALUE 'I'.
 77  F22-KEY                        PIC X VALUE '¢'.
 77  F23-KEY                        PIC X VALUE '.'.
 77  F24-KEY                        PIC X VALUE '<'.
 77  PA1-KEY                        PIC X VALUE '%'.
 77  PA2-KEY                        PIC X VALUE '>'.
 77  PA3-KEY                        PIC X VALUE ','.
 77  CLR-KEY                        PIC X VALUE '_'.
 77  TST-KEY                        PIC X VALUE '0'.
 77  NOAID-1                        PIC X VALUE '-'.
 77  NOAID-2                        PIC X VALUE 'Y'.
*
 01  TR-CTL-AREA.
     05  TR-FUNCTION-KEYS           PIC X(02).
     05  TR-TERMINAL-ID             PIC X(10).
     05  TR-FORMAT-NAME             PIC X(10).
*
 01  MAJ-MIN.
     05  MAJ                        PIC X(02).
     05  MIN                        PIC X(02).
*
 01  WS-CTL.
     05  CMD-KEY                    PIC X(02).
     05  FILLER                     PIC X(10).
     05  RCD-FMT                    PIC X(10).
*
 01  DATA-3270-STRCT.
     05  AID-BYTE                   PIC X.      15
     05  CURPOS.
         10  CURPOS-1               PIC X.
         10  CURPOS-2               PIC X.
     05  DATA-3270                  PIC X(253).
*
 01  HEX-00-BINARY                  PIC 999 COMP-4 VALUE 00.
 01  HEX-00-R  REDEFINES  HEX-00-BINARY.
     05  FILLER                     PIC X.
     05  HEX00                      PIC X.
*
 01  HEX-11-BINARY                  PIC 999 COMP-4 VALUE 17.
 01  HEX-11-R REDEFINES  HEX-11-BINARY.
     05  FILLER                     PIC X.
     05  HEX-SBA                    PIC X.
*
 01  HEX-4A-BINARY                  PIC 999 COMP-4 VALUE 74.
 01  HEX-4A-R  REDEFINES  HEX-4A-BINARY.
     05  FILLER                     PIC X.
     05  PF22-KEY                   PIC X.
*
 01  HEX-6A-BINARY                  PIC 999 COMP-4 VALUE 106.
 01  HEX-6A-R  REDEFINES  HEX-6A-BINARY.
     05  FILLER                     PIC X.
     05  HEX-6A                     PIC X.
*
 PROCEDURE DIVISION.
*
 MAIN-LINE SECTION.
*
 MAIN-LINE-ROUTINE.                         12
*
     PERFORM OPEN-FILES THRU EXIT-OPEN-FILES.
     PERFORM ACQ-DEV THRU EXIT-ACQ-DEV.
     PERFORM MAIN-ROUTINE THRU EXIT-MAIN-ROUTINE.
     PERFORM CLOSE-FILES THRU EXIT-CLOSE-FILES.
     PERFORM STOP-RUN THRU EXIT-STOP-RUN.
*
 OPEN-FILES.
     OPEN I-O    ICF01
          I-O    SHOW.
 EXIT-OPEN-FILES.
     EXIT.
*
 ACQ-DEV.                                   3
     ACQUIRE 'CICSDEV' FOR ICF01.
     IF MAJ NOT = '00'
       THEN
         PERFORM DSP-ERR-INFO THRU EXIT-DSP-ERR-INFO
         PERFORM CLOSE-FILES THRU EXIT-CLOSE-FILES.
 EXIT-ACQ-DEV.
     EXIT.
*
 DSP-ERR-INFO.
     DISPLAY '  MAJ/MIN ==>' MAJ-MIN.
     DISPLAY '  ICF STATUS ==>' ICF-STATUS.
 EXIT-DSP-ERR-INFO.
     EXIT.
*
 MAIN-ROUTINE.                               13
     PERFORM READ-RU THRU EXIT-READ-RU.
     PERFORM SEND-CLR THRU EXIT-SEND-CLR.
     PERFORM READ-RU THRU EXIT-READ-RU.
     PERFORM SEND-ID01 THRU EXIT-SEND-ID01.
     PERFORM READ-RU THRU EXIT-READ-RU.    8
     PERFORM READ-CHG-DIR THRU EXIT-READ-CHG-DIR.
     PERFORM SEND-F11 THRU EXIT-SEND-F11.
     PERFORM READ-RU THRU EXIT-READ-RU.
 EXIT-MAIN-ROUTINE.
     EXIT.
***************************************************************
*            READ 3270 DATA STREAM IN                         *
*                                                             *
* IN THIS EXAMPLE WE CHECK THE MAJOR MINOR RETURN CODE ONLY.  *
* IN A PRODUCTION ENVIRONMENT THE INPUT DATA FLOW CONTROL     *
* FIELD WOULD BE VERIFIED ALSO. WHEN A READ IS PERFORMED IF   *
* ANOTHER DATA FLOW IS ENCOUNTER, SOME OTHER ACTION MAY BE    *
* NECESSARY. IN THIS CASE AN LU TO LU FLOW IS EXPECTED SO A   *
* HEX F0 WILL BE IN THE I-FLOW FIELD IN THE INPUT HEADER.     *
***************************************************************
*
 READ-RU.
     MOVE SPACES TO INPUT-BUFFER.
     READ ICF01 TERMINAL IS 'CICSDEV'.
     IF MAJ-MIN NOT < '0004'
       THEN
         PERFORM DSP-ERR-INFO THRU EXIT-DSP-ERR-INFO
         PERFORM END-SESS THRU EXIT-END-SESS
         PERFORM REL-DEV THRU EXIT-REL-DEV
         PERFORM CLOSE-FILES THRU EXIT-CLOSE-FILES
         PERFORM STOP-RUN THRU EXIT-STOP-RUN.
     PERFORM DSP-IN-DATA THRU EXIT-DSP-IN-DATA.
 EXIT-READ-RU.
     EXIT.
*
 READ-CHG-DIR.
     MOVE SPACES TO INPUT-BUFFER.
     READ ICF01 TERMINAL IS 'CICSDEV'.
     IF MAJ-MIN NOT = '0300'
       THEN
         PERFORM DSP-ERR-INFO THRU EXIT-DSP-ERR-INFO
         PERFORM END-SESS THRU EXIT-END-SESS
         PERFORM REL-DEV THRU EXIT-REL-DEV
         PERFORM CLOSE-FILES THRU EXIT-CLOSE-FILES
         PERFORM STOP-RUN THRU EXIT-STOP-RUN.
     PERFORM DSP-IN-DATA THRU EXIT-DSP-IN-DATA.
 EXIT-CHG-DIR.
     EXIT.
*
```

```
 DSP-IN-DATA.                  14
     MOVE SPACES TO DSP-REC.
     MOVE L01-IN TO SHOW-RECORD.
     PERFORM CONV-HEX THRU EXIT-CONV-HEX
         VARYING CONV-INDX FROM 1 BY 1 UNTIL CONV-INDX = 70.
     WRITE DSP-REC FORMAT IS 'DSP1919'.
     READ SHOW.
 EXIT-DSP-IN-DATA.
     EXIT.
*
 CONV-HEX.
     IF L1-SUB(CONV-INDX) < ' '
         THEN MOVE '?' TO L1-SUB(CONV-INDX).
 EXIT-CONV-HEX.
     EXIT.
*
 SEND-CLR.
     MOVE 21 TO OUTPUT-LENGTH.
     MOVE HEX00 TO O-COMMAND.
     MOVE '0' TO O-FLOW.
     MOVE SPACES TO OUTPUT-DATA, DATA-3270-STRCT.
     MOVE CLR-KEY TO AID-BYTE.
     MOVE '  ' TO CURPOS.
     PERFORM SEND-3270 THRU EXIT-SEND-3270.
 EXIT-SEND-CLR.
     EXIT.
*
 SEND-ID01.
     MOVE 27 TO OUTPUT-LENGTH.
     MOVE HEX00 TO O-COMMAND.
     MOVE '0' TO O-FLOW.
     MOVE SPACES TO OUTPUT-DATA, DATA-3270-STRCT
     MOVE ENTER-KEY TO AID-BYTE.
     MOVE ' D' TO CURPOS.
     MOVE 'ID01' TO DATA-3270.
     PERFORM SEND-3270 THRU EXIT-SEND-3270.
 EXIT-SEND-ID01.
     EXIT.
*
 SEND-F11.
     MOVE 23 TO OUTPUT-LENGTH.
     MOVE HEX00 TO O-COMMAND.
     MOVE '0' TO O-FLOW.
     MOVE SPACES TO OUTPUT-DATA, DATA-3270-STRCT.
     MOVE F11-KEY TO AID-BYTE.
     MOVE SPACES TO CURPOS.
     PERFORM SEND-3270 THRU EXIT-SEND-3270.
 EXIT-SEND-F11.
     EXIT.
*
 SEND-3270.                              17
     MOVE DATA-3270-STRCT TO OUTPUT-DATA.
     WRITE ICF-REC FORMAT IS '$$SEND'.
     IF MAJ-MIN NOT = '0000'
       THEN
         PERFORM DSP-ERR-INFO THRU EXIT-DSP-ERR-INFO
         PERFORM END-SESS THRU EXIT-END-SESS
         PERFORM REL-DEV THRU EXIT-REL-DEV
         PERFORM CLOSE-FILES THRU EXIT-CLOSE-FILES
         PERFORM STOP-RUN THRU EXIT-STOP-RUN.
 EXIT-SEND-3270.
     EXIT.
*
```

```
 REL-DEV.
     MOVE ZEROS TO OUTPUT-LENGTH.
     WRITE ICF-REC FORMAT IS '$$SENDET'.
     IF MAJ-MIN NOT = '0000'
       THEN
         PERFORM DSP-ERR-INFO THRU EXIT-DSP-ERR-INFO
         PERFORM CLOSE-FILES THRU EXIT-CLOSE-FILES
         PERFORM STOP-RUN THRU EXIT-STOP-RUN.
 EXIT-REL-DEV.
     EXIT.
*
 END-SESS.
     WRITE ICF-REC FORMAT IS '$$EOS'.
     IF MAJ-MIN NOT = '0000'
       THEN
         PERFORM DSP-ERR-INFO THRU EXIT-DSP-ERR-INFO
         PERFORM REL-DEV THRU EXIT-REL-DEV
         PERFORM CLOSE-FILES THRU EXIT-CLOSE-FILES
         PERFORM STOP-RUN THRU EXIT-STOP-RUN.
 EXIT-END-SESS.
     EXIT.
*
 CLOSE-FILES.
     DISPLAY 'CLOSING FILES'.
     CLOSE ICF01.
     CLOSE SHOW.
 EXIT-CLOSE-FILES.
     EXIT.
*
 STOP-RUN.
     STOP RUN.
 EXIT-STOP-RUN.
     EXIT.
```

Appendix E. Program Examples

This appendix contains two examples of an item-inquiry application. The application consists of Program A, which is the AS/400 program, and Program B, which is the program for the remote host system. For an SNA 3270 program example, see Appendix D.

“Example 1: AS/400 System to System/370 System (CICS/VS)” on page E-2 illustrates communications programming between the AS/400 system and a System/370 system operating under CICS/VS. Program A is a ILE COBOL/400 program with system-supplied formats; Program B is a System/370 COBOL CICS/VS program.

“Example 2: AS/400 System to System/370 System (IMS/VS)” on page E-17 illustrates communications programming between the AS/400 system and a System/370 system operating under IMS/VS. Program A is an ILE RPG/400 program with data description specifications (DDS); Program B is a System/370 COBOL IMS/VS program.

“Example 3: AS/400 System to System/370 System (CICS/VS)” on page E-27 illustrates communications programming between the AS/400 system and a System/370 system operating under CICS/VS. Program A is a ILE C/400 program with system-supplied formats; Program A is a System/370 COBOL CICS/VS program.

Before running the program examples, create a line description, a controller description, and device descriptions on the AS/400 system. Sample commands follow:

```
CRTLINSDLC
  LIND(XLINE)
  RSRCNAME(LIN011)
  ROLE(*SEC)

CRTCTLHOST
  CTLD(XCTLR)
  LINKTYPE(*SDLC)
  APPN(*NO)
  LINE(XLINE)
  STNADR(C1)

CRTDEVSNUF
  DEVD(XDEV)
  LOCADR(06)
  RMTLOCNAME(CICSLOC)
  CTL(XCTLR)
  PGMSTRRQS(*NO)
  APPID(CICS)
  HOST(*CICS)

CRTDEVSNUF
  DEVD(YDEV)
  LOCADR(0A)
  RMTLOCNAME(IMSLOC)
  CTL(XCTLR)
  PGMSTRRQS(*NO)
  APPID(XAIMS)
  HOST(*IMS)
```

The item-inquiry application for both “Example 1: AS/400 System to System/370 System (CICS/VS)” and “Example 2: AS/400 System to System/370 System (IMS/VS)” is shown in Figure E-1.

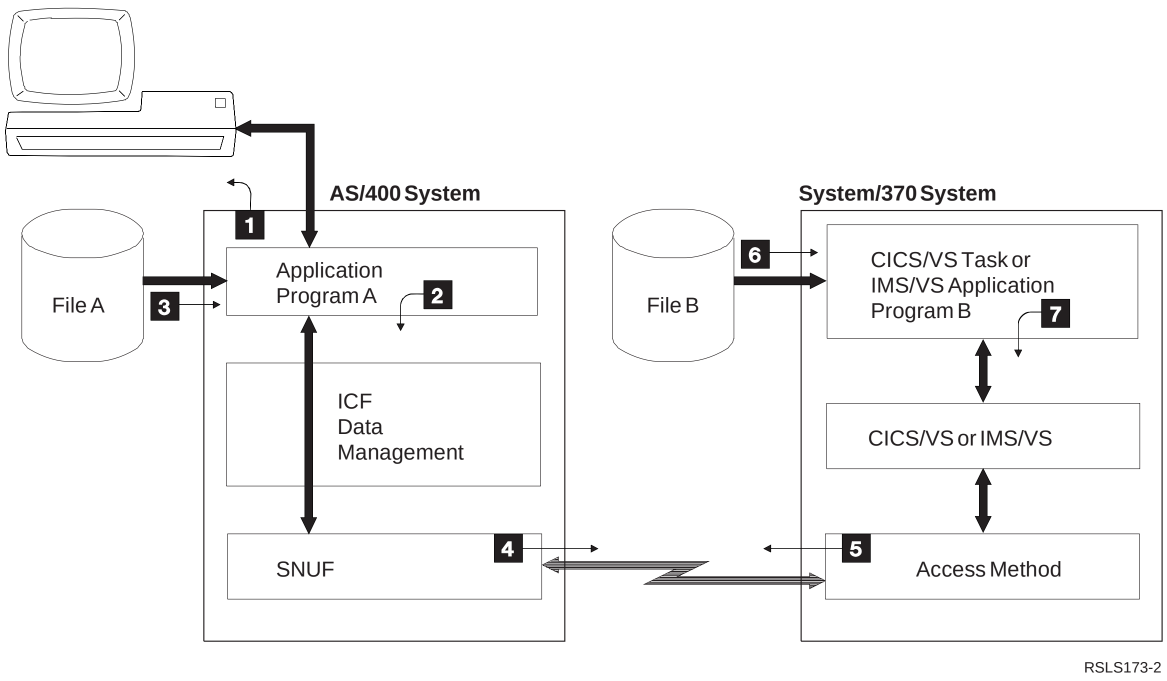


Figure E-1. Item-Inquiry Application

In Figure E-1:

- Application Program A (in the AS/400 system) shows a prompt asking an operator to enter an item number requesting information about the item **1**.
- When the operator enters the item number, Program A reads the number and searches File A (the local file) for the item **3**.
- If the item is found in the local file, Program A shows information about the item **1**.
- If the item is not found on the local file, Program A uses SNUF to send the item number to the host system **2** and **4**.
- Program B (in the host system) uses the item number to search the remote file for the item **6**.
- If the item is found in the remote file, Program B sends information about the item to Program A **7** and **5**. If the item is not found in the remote file, Program B sends the characters `***`.
- If Program A receives information about the item, it shows that information. If it receives the characters ***, it shows the message `ITEM NUMBER NOT FOUND` **1**.

Example 1: AS/400 System to System/370 System (CICS/VS)

The following example consists of an AS/400 COBOL program with system-supplied formats, talking to a System/370 COBOL CICS/VS program.

Not all programming considerations or techniques are illustrated in this example. You should review the example before you begin application design and coding.

ILE COBOL/400 Program for the AS/400 System (Program A)

The following program (PROGACOB) is used on the AS/400 system.

When PROGACOB is called, a display is presented. This display contains a single inquiry line that is 23 bytes long. Type an item number on the inquiry line and press the Enter key to begin searching. The local database (FILEA) is searched, and if a matching item is found, up to four item quantities are displayed.

If a matching item number is not in the local file, then a request is built and sent to the host system. The program, ICII, is started to obtain the matching item number and quantities from the remote system database; the matching item number and its associated quantities are returned to PROGACOB, which displays the quantities that are available.

DDS Source: The DDS source for DISPFILE is illustrated in Figure E-2. The other files are either program-described (FILEA and PRINTFILE) or default (QICDMF) and, therefore, require no DDS.

```
A****************************************************************
A*                                                              *
A*     LOCAL DISPLAY FILE                                       *
A*                                                              *
A****************************************************************
A*
A                                      CA01
A                                      CA07
A          R WSFILEIN
A            ITMNUM        23   I  2 10
A  01                           O  4 10'INVALID ITEM NUMBER ENTERED'
A  02                           O  4 10'ITEM NUMBER NOT FOUND'
A                               O 12 10'PRESS CMD KEY 7 TO TERMINATE'
A          R WSFILEOT
A            ITMNUM        23   O  2 10
A            QTY1           6  00  4 12
A            QTY2           6  00  5 12
A            QTY3           6  00  6 12
A            QTY4           6  00  7 12
A            MSG           80   O  8 10
A            RETCOD         4   O 10 10
A            REASON        30   O 11 10
A            FILLER        95   O 12 10
A                               O 14 10'PRESS CMD KEY 7 TO TERMINATE OR'
A                               O 15 10'CMD KEY 1 FOR ANOTHER INQUIRY'
```

Figure E-2. DDS Source for DISPFILE

Program Device Entry Definition: The command needed to define the program device entry is:

```
ADDICFDEVE FILE(*LIBL/QICDMF)
           PGMDEV(SNUFDEVICE)
           RMTLOCNAME(CICSLOC)
           CMNTYPE(*SNUF)
           DEV(XDEV)
           APPID(CICS)
           HOST(*DEVD) or (*CICS)
```

ILE COBOL/400 Program: Figure E-3 shows the ILE COBOL/400 program PROGACOB for the AS/400 system.

```
******************************************************************
*                                                                *
*    PROGACOB - ITEM INQUIRY WRITTEN IN COBOL                    *
*                                                                *
******************************************************************
      *
       IDENTIFICATION DIVISION.
      *
      ************************************************************
      *                                                          *
      *    PROGACOB - ITEM INQUIRY WRITTEN IN COBOL              *
      *                                                          *
      ************************************************************
       PROGRAM-ID. PROGACOB.
       ENVIRONMENT DIVISION.
      *
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  IBM-AS400.
       OBJECT-COMPUTER.  IBM-AS400.
      *
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT QICDMF
              ASSIGN TO WORKSTATION-QICDMF-SI
              ORGANIZATION IS TRANSACTION
              FILE STATUS IS STATUS-IND MAJ-MIN
              CONTROL-AREA IS COMM-CONTROL-AREA.
      *
           SELECT DISPFILE
              ASSIGN TO WORKSTATION-DISPFILE
              ORGANIZATION IS TRANSACTION
              FILE STATUS IS WS-FS
              CONTROL-AREA IS WS-CONTROL-AREA.
      *
           SELECT FILEA ASSIGN TO DISK-FILEA
              ORGANIZATION IS INDEXED
              ACCESS IS RANDOM
              RECORD KEY IS FILEA-NUMBER.
      *
           SELECT PRINTFILE ASSIGN TO PRINTER-PRINTFILE.
      *
       DATA DIVISION.
       FILE SECTION.
       FD  QICDMF
           LABEL RECORDS ARE OMITTED.
       01  COMMREC.
         05  COMMUNICATION-RECORD          PIC X(256).
       FD  DISPFILE, LABEL RECORDS ARE OMITTED.
       01  DISPREC.
           03 ITEM-NUMBER              PIC X(23).
           03 QTY-1                    PIC 9(6).
           03 QTY-2                    PIC 9(6).
           03 QTY-3                    PIC 9(6).
           03 QTY-4                    PIC 9(6).
           03 MSG                      PIC X(80).
           03 RETURN-CODE              PIC X(4).
           03 REASON-WHY               PIC X(30).
           03 FILL1                    PIC X(95).
      *
```

Figure E-3 (Part 1 of 6). COBOL Program A for the AS/400 System

```
 FD  FILEA, LABEL RECORDS ARE STANDARD.
 01  FILEAREC.
   03  FILEA-RECORD.
     05  FILEA-NUMBER            PIC X(23).
     05  FILLER                  PIC X(3).
     05  FILEA-QTY-1             PIC 9(6).
     05  FILEA-QTY-2             PIC 9(6).
     05  FILEA-QTY-3             PIC 9(6).
     05  FILEA-QTY-4             PIC 9(6).
*
 FD  PRINTFILE, LABEL RECORDS ARE OMITTED.
 01  PRINT-RECORD                PIC X(132).
*
 WORKING-STORAGE SECTION.
 77  ICF-SESSION                 PIC X(10) VALUE "SNUFDEVICE".
 77  SAVE-ITEM-NUMBER            PIC X(23).
 77  STATUS-IND                  PIC X(2).
 77  WS-FS                       PIC X(2).
*
 01  COMM-CONTROL-AREA.
    03  FILLER                   PIC X(2).
    03  PGM-DEV-NAME             PIC X(10).
    03  RCD-FMT-NAME             PIC X(10).
*
 01  WS-CONTROL-AREA.
   03  CMD-KEY                  PIC X(2).
          88  CMD-KEY-1      VALUE "01".
          88  CMD-KEY-7      VALUE "07".
   03  FILLER                   PIC X(10).
   03  RCD-FMT                  PIC X(10).
*
 01  SCREEN-INDICATORS.
     03  I01                     PIC 1 VALUE ZERO, INDICATOR 01.
     03  I02                     PIC 1 VALUE ZERO, INDICATOR 02.
     03  I03                     PIC 1 VALUE ZERO, INDICATOR 03.
*
 01  ACQUIRE-INDICATOR.
     03  I04                     PIC 1 VALUE ZERO, INDICATOR 01.
*
 01  EVOKE-RECORD.
     03  PROCEDURE-NAME          PIC X(8) VALUE "ICII    ".
     03  PASSWORD                PIC X(8).
     03  USER-ID                 PIC X(8).
     03  LIBRARY-NAME            PIC X(8).
     03  FILLER                  PIC X(20).
     03  DATA-LENGTH             PIC X(4) VALUE "0023".
     03  ICF-ITEM-NUMBER-OUT     PIC X(23).
*
 01  ICF-RECORD-IN.
     03  ICF-RECORD-CHECK.
       05  FIRST-3-CHARACTERS    PIC X(3).
       05  REST-OF-DATA          PIC X(253).
*
     03  ICF-RECORD-OK REDEFINES ICF-RECORD-CHECK.
       05  FILLER                PIC X(32).
       05  ICF-ITEM-NUMBER-IN    PIC X(23).
       05  FILLER                PIC X(145).
       05  ICF-QTY-1             PIC 9(6).
       05  ICF-QTY-2             PIC 9(6).
       05  ICF-QTY-3             PIC 9(6).
       05  ICF-QTY-4             PIC 9(6).
       05  FILLER                PIC X(32).
*
```

Figure E-3 (Part 2 of 6). COBOL Program A for the AS/400 System

```
 01  ICF-RECORD-OUT.
    03  RECORD-LENGTH            PIC 9(4).
    03  THE-RECORD               PIC X(256).
*
 01  MAJ-MIN.
     03 MAJOR-RETURN-CODE          PIC X(2).
     03 MINOR-RETURN-CODE          PIC X(2).
*
 01  PRINT-CODES.
     03  FILLER                  PIC X(14) VALUE "RETURN CODE ".
     03  PRINT-RETURN-CODE       PIC X(4).
     03  FILLER                  PIC X(11) VALUE " OPCODE IS ".
     03  OPCODE                  PIC X(6).
     03  FILLER                  PIC X(11) VALUE " DATA SENT ".
     03  PRINT-ITEM-NUMBER       PIC X(23).
*
 PROCEDURE DIVISION.
*
************************************************************
*                                                          *
*    INITIALIZATION                                        *
*                                                          *
************************************************************
 OPEN-FILES.
     OPEN I-O DISPFILE, QICDMF.
     OPEN OUTPUT PRINTFILE.
     OPEN INPUT FILEA.
     MOVE B"0" TO I01, I02, I03, I04.
     MOVE SPACES TO DISPREC.
*
************************************************************
*                                                          *
*    DISPLAY SCREEN REQUESTING ITEM NUMBER.  IF CMD 7,     *
*    GO TO CLOSE FILES.  SET UP INDICATORS TO DISPLAY      *
*    ERRORS IF ITEM NUMBER IS SPACE OR ZEROS               *
*                                                          *
************************************************************
 ITEM-INQUIRY.
     IF I03 = B"1"
        MOVE B"0" TO I03
        WRITE DISPREC
          FORMAT IS "WSFILEOT"
        READ DISPFILE
          INDICATORS ARE SCREEN-INDICATORS
        IF CMD-KEY-7
          GO TO CLOSE-FILES.
*    MUST BE CMD-KEY-1
     WRITE DISPREC
        FORMAT IS "WSFILEIN"
        INDICATORS ARE SCREEN-INDICATORS.
     READ DISPFILE
        FORMAT IS "WSFILEIN".
     MOVE B"0" TO I01, I02, I03.
     IF CMD-KEY-7
       GO TO CLOSE-FILES.
     IF ITEM-NUMBER = SPACES OR ITEM-NUMBER = ZEROS
        MOVE B"1" TO I01
        GO TO ITEM-INQUIRY.
```

Figure E-3 (Part 3 of 6). COBOL Program A for the AS/400 System

```
      *
      ************************************************************
      *                                                          *
      *    READ LOCAL FILE 'FILEA' FOR REQUESTED ITEM NUMBER.    *
      *    IF ITEM IS FOUND LOCALLY, DISPLAY ITEM INFORMATION.   *
      *    IF ITEM IS NOT FOUND LOCALLY, INQUIRE OF 'ITEMBCOB'   *
      *    USING ICF.                                            *
      *                                                          *
      ************************************************************
       READ-FILEA-FILE.
           MOVE SPACES TO FILEA-RECORD.
           MOVE ITEM-NUMBER TO FILEA-NUMBER.
           READ FILEA,
             INVALID KEY
               GO TO ICF.
           MOVE FILEA-QTY-1 TO QTY-1.
           MOVE FILEA-QTY-2 TO QTY-2.
           MOVE FILEA-QTY-3 TO QTY-3.
           MOVE FILEA-QTY-4 TO QTY-4.
           GO TO ITEM-INQUIRY.
      *
      ************************************************************
      *                                                          *
      *    ACQUIRE ICF-SESSION (SNUFDEVICE), IF NOT ACQUIRED.    *
      *                                                          *
      ************************************************************
       ICF.
           IF I04 = B"0"
             MOVE B"1" TO I04
             ACQUIRE ICF-SESSION FOR QICDMF
             MOVE "ACQ" TO OPCODE
             PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END
             IF I03 = B"1"
                GO TO ITEM-INQUIRY.
      *
      ************************************************************
      *                                                          *
      *    EVOKE 'ICII' AT HOST.                                 *
      *                                                          *
      ************************************************************
           MOVE ITEM-NUMBER TO ICF-ITEM-NUMBER-OUT.
           WRITE COMMREC FROM EVOKE-RECORD
              FORMAT IS "$$EVOK".
           MOVE "EVOK" TO OPCODE.
           PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
           IF I03 = B"1"
              PERFORM SEND-EOS
              GO TO ITEM-INQUIRY.
      *
```

Figure E-3 (Part 4 of 6). COBOL Program A for the AS/400 System

```
************************************************************
*                                                          *
*    GET INPUT FROM HOST.                                  *
*                                                          *
************************************************************
     MOVE SPACES TO ICF-RECORD-IN.
     READ QICDMF RECORD INTO ICF-RECORD-IN.
     MOVE "GET" TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF I03 = B"1"
        PERFORM SEND-EOS
        GO TO ITEM-INQUIRY.
*
************************************************************
*                                                          *
*    SEND END OF TRANSACTION.                              *
*                                                          *
************************************************************
     MOVE SPACES TO DISPREC.
     MOVE 0 TO RECORD-LENGTH.
     WRITE COMMREC FROM ICF-RECORD-OUT
        FORMAT IS "$$SENDET".
     MOVE "SENDET" TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF I03 = B"1"
        PERFORM SEND-EOS
        GO TO ITEM-INQUIRY.
*
************************************************************
*                                                          *
*    CHECK FOR ERROR MESSAGE (3 ASTERISKS).                *
*    IF ITEM NUMBER IS NOT FOUND, DISPLAY MESSAGE 'ITEM    *
*    NUMBER NOT FOUND' TO THE SCREEN.                      *
*    IF THE ITEM IS FOUND, DISPLAY THE INVENTORY INFOR.    *
*                                                          *
************************************************************
     MOVE B"0" TO I01, I02, I03.
     MOVE ITEM-NUMBER TO SAVE-ITEM-NUMBER.
     MOVE SPACES TO DISPREC.
     IF FIRST-3-CHARACTERS = "***"
        MOVE B"1" TO I02
     ELSE
        MOVE B"1" TO I03
        MOVE ICF-ITEM-NUMBER-IN TO ITEM-NUMBER
        MOVE ICF-QTY-1 TO QTY-1
        MOVE ICF-QTY-2 TO QTY-2
        MOVE ICF-QTY-3 TO QTY-3
        MOVE ICF-QTY-4 TO QTY-4.
     GO TO ITEM-INQUIRY.
```

Figure E-3 (Part 5 of 6). COBOL Program A for the AS/400 System

```
*
***********************************************************
*                                                         *
*    CLOSE FILES AND END JOB.                             *
*                                                         *
***********************************************************
 CLOSE-FILES.
     CLOSE QICDMF, DISPFILE, FILEA, PRINTFILE.
     STOP RUN.
*
 SEND-EOS.
     MOVE B"0" TO I04.
     MOVE SPACES TO DISPREC.
     WRITE COMMREC
        FORMAT IS "$$EOS".
     MOVE "EOS" TO OPCODE.
     MOVE MAJ-MIN TO PRINT-RETURN-CODE.
     WRITE PRINT-RECORD FROM PRINT-CODES
        AFTER ADVANCING 2 LINES.
*
***********************************************************
*                                                         *
*    CHECK RETURN CODE.                                   *
*                                                         *
***********************************************************
 CHECK-RETURN-CODE.
     IF MAJOR-RETURN-CODE < "04"
        GO TO CHECK-RETURN-CODE-END.
*
     MOVE ITEM-NUMBER TO PRINT-ITEM-NUMBER.
     MOVE MAJ-MIN TO PRINT-RETURN-CODE.
     WRITE PRINT-RECORD FROM PRINT-CODES
        AFTER ADVANCING 2 LINES.
     MOVE SPACES TO RETURN-CODE.
     IF MAJOR-RETURN-CODE = "04"
          MOVE MAJ-MIN TO RETURN-CODE
          MOVE "OUTPUT EXCEPTION" TO REASON-WHY
          READ QICDMF RECORD INTO ICF-RECORD-CHECK
          MOVE ICF-RECORD-CHECK TO MSG
          MOVE B"1" TO I03
     ELSE
          IF MAJOR-RETURN-CODE = "82"
             MOVE MAJ-MIN TO RETURN-CODE
             MOVE "UNABLE TO ACQUIRE" TO REASON-WHY
             MOVE B"1" TO I03
             MOVE B"0" TO I04
          ELSE
             IF MAJOR-RETURN-CODE > "04"
             MOVE MAJ-MIN TO RETURN-CODE
             MOVE B"1" TO I03.
 CHECK-RETURN-CODE-END.
```

Figure E-3 (Part 6 of 6). COBOL Program A for the AS/400 System

CICS/VS Program Used by the Host System (Program B)

The program in Figure E-4 is used by the host system to communicate with the AS/400 system.

```
MEMBER NE  ICII
       IDENTIFICATION DIVISION.
           SKIP3
      ************************************************************
      *                                                          *
      *    ICII       INVENTORY INQUIRY                          *
      *                                                          *
      ************************************************************
           SKIP3
         PROGRAM-ID.  ICII.
           SKIP1
           SKIP3
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
           SKIP1
         SOURCE-COMPUTER.  IBM-370-155.
         OBJECT-COMPUTER.  IBM-370-155.
           EJECT
       DATA DIVISION.
           SKIP3
       WORKING-STORAGE SECTION.
           SKIP1
       77  FUNCTION                PIC X(4).
       77  PSB-NAME                PIC X(8)        VALUE 'MSCTG005'.
       77  NORESP                  PIC X(1)        VALUE LOW-VALUES.
       77  PCBOK                   PIC X(2)        VALUE SPACES.
       77  TDI-LENGTH              PIC S9(4) COMP  VALUE +256.
       77  TDO-LENGTH              PIC S9(4) COMP  VALUE +256.
       77  ABEND-CODE              PIC X(4)        VALUE SPACES.
           EJECT
      ***********************************************************
      *                                                         *
      *    TRANSACTION DATA RECEIVED FROM IBM AS/400 SYSTEM     *
      *                                                         *
      ***********************************************************
           SKIP1
       01  TRANS-DATA-IN.
           05  TDI-PGMID           PIC X(4).
           05  TDI-FILL1           PIC X(1).
           05  TDI-MSINITEM        PIC X(23).
           SKIP3
      ************************************************************
      *                                                          *
      *    TRANSACTION DATA SENT BACK TO IBM AS/400 SYSTEM       *
      *                                                          *
      ************************************************************
           SKIP1
       01  TRANS-DATA-OUT.
           05  TDO-FILL1           PIC X(2)        VALUE SPACES.
           05  TDO-FILLX           PIC X(30)       VALUE SPACES.
           05  TDO-MSINITEM        PIC X(23).
           05  TDO-FILL2           PIC X(145)      VALUE SPACES.
           05  TDO-MSLCHAND1       PIC S9(6).
           05  TDO-MSLCHAND2       PIC S9(6).
           05  TDO-MSLCHAND3       PIC S9(6).
           05  TDO-MSLCHAND4       PIC S9(6).
```

Figure E-4 (Part 1 of 7). CICS/VS Program Used by the Host System

```
MEMBER NE  ICII
           05  TDO-FILL3           PIC X(32)       VALUE SPACES.
           SKIP3
       01  ERROR-MESSAGE REDEFINES TRANS-DATA-OUT.
           05  EM-MESSAGE          PIC X(51).
           05  EM-FILLER           PIC X(205).
           EJECT
      ************************************************************
      *                                                          *
      *    DL/I INVENTORY DATABASE SEGMENT INPUT AREA            *
      *                                                          *
      ************************************************************
           SKIP1
       01  SEGMENT-MSININ01-IN.
           05  MSINITEM            PIC X(23).
           05  MSINBCOD            PIC X(2).
           05  MSINCOST            PIC S9(7)       COMP-3.
           05  MSINDESC            PIC X(10).
           SKIP3
       01  SEGMENT-MSLCIN01-IN.
           05  MSLCLOCD            PIC X(2).
           05  MSLCHAND            PIC S9(9)       COMP-3.
           05  MSLCALOC            PIC S9(9)       COMP-3.
           05  MSLCCLOC            PIC S9(9)       COMP-3.
           05  MSLCBORD            PIC S9(9)       COMP-3.
           05  MSLCWORK            PIC S9(9)       COMP-3.
           EJECT
      ************************************************************
      *                                                          *
      *    DL/I INVTY DATABASE SEGMENT SEARCH ARGUMENTS (SSA'S)  *
      *                                                          *
      ************************************************************
           SKIP1
       01  INVTY-SEGMENT-SSA.
           05  FILLER          PIC X(19)   VALUE 'MSININ01(MSINITEM ='.
           05  SSA-MSINITEM    PIC X(23).
           05  FILLER          PIC X(1)    VALUE ')'.
           SKIP3
       01  QTYS-SEGMENT-SSA.
       05  FILLER              PIC X(9)    VALUE 'MSLCIN01 '.
           EJECT
      ************************************************************
      *                                                          *
      *    MISCELLANEOUS WORKING-STORAGE AREAS.                  *
      *                                                          *
      ************************************************************
           SKIP1
       01  DLI-FUNCTIONS.
           05  GU              PIC X(4)    VALUE 'GU  '.
           05  GNP             PIC X(4)    VALUE 'GNP '.
           05  PCB             PIC X(4)    VALUE 'PCB '.
           05  TERM            PIC X(4)    VALUE 'TERM'.
           SKIP3
```

Figure E-4 (Part 2 of 7). CICS/VS Program Used by the Host System

```
 01  MSG-NOT-FOUND.
     05  FILLER          PIC X(11)   VALUE '***ERROR***'.
     05  FILLER          PIC X(6)    VALUE ' ITEM '.
     05  MSG-MSINITEM    PIC X(23).
     05  FILLER          PIC X(11)   VALUE ' NOT FOUND.'.
     SKIP3
 01  DLI-SWITCH          PIC 9       VALUE 0.
     88  QTYS-OK                     VALUE 0.
     88  QTYS-UIB-ERROR              VALUE 1.
     EJECT
 LINKAGE SECTION.
     SKIP1
 01  BLL-CELLS.
     05  FILLER          PIC 9(8)    COMP.
     05  DLI-UIB-PTR     PIC 9(8)    COMP.
     05  DLI-PCB-PTRS    PIC 9(8)    COMP.
     05  PCB1-PTR.
       10  FILLER        PIC 9(8)    COMP.
     EJECT
     COPY DLIUIB.
     SKIP3
 01  PCB-PTRS.
     05  B-PCB1-PTR      PIC S9(8)   COMP.
     SKIP3
 01  PCB1.
     05  PCB1-DBD-NAME   PIC X(8).
     05  PCB1-SEG-LEVEL  PIC X(2).
     05  PCB1-STATUS     PIC X(2).
       88  INVENTORY-SEGMENT-FOUND   VALUE '  '.
       88  END-OF-CHAIN              VALUE 'GE'.
       88  INVENTORY-DOESNT-EXIST    VALUE 'GE'.
       88  QUANTITY-SEGMENT-FOUND    VALUE '  '.
     05  PCB1-PROC-OPTN  PIC X(1).
     05  PCB1-FILL1      PIC S9(5)   COMP.
     05  PCB1-SEG-NAME   PIC X(8).
     05  PCB1-LENG-KFBA  PIC S9(5)   COMP.
     05  PCB1-NUMBOSSEG  PIC S9(5)   COMP.
     05  PCB1-KEY-FBA    PIC X(256).
     EJECT
************************************************************
*                                                          *
*    MAIN LINE ROUTINE.                                    *
*                                                          *
************************************************************
     SKIP1
 PROCEDURE DIVISION.
     SKIP1
     PERFORM GET-PSB.
     IF PCB1-DBD-NAME = 'MSDPIN01'
       PERFORM GET-PSB.
       MOVE TDI-MSINITEM TO TDO-MSINITEM, SSA-MSINITEM
       PERFORM GET-INVTY
       IF INVENTORY-SEGMENT-FOUND
         PERFORM GET-QTYS-ON-HAND THRU GET-QTYS-EXIT
         IF QTYS-OK
         PERFORM SEND-DATA
       ELSE
         MOVE 'QTYU' TO ABEND-CODE
         PERFORM CICS-ABEND
```

Figure E-4 (Part 3 of 7). CICS/VS Program Used by the Host System

```
         ELSE
           IF INVENTORY-DOESNT-EXIST
             MOVE TDI-MSINITEM  TO MSG-MSINITEM
             MOVE SPACES        TO EM-FILLER
             MOVE MSG-NOT-FOUND TO EM-MESSAGE
             PERFORM SEND-DATA
           ELSE
             MOVE 'ITYP' TO ABEND-CODE
             PERFORM CICS-ABEND
         ELSE
           MOVE 'INVD' TO ABEND-CODE
           PERFORM CICS-ABEND.
         SKIP3
         PERFORM RELEASE-PSB.
         SKIP1
         EXEC CICS RETURN END-EXEC.
         SKIP1
         GOBACK.
         EJECT
************************************************************
*                                                          *
*     CLOSED SUBROUTINES.                                  *
*                                                          *
************************************************************
         SKIP1
     GET-PSB.
         SKIP1
         MOVE PCB TO FUNCTION.
         CALL 'CBLTDLI' USING FUNCTION, PSB-NAME, DLI-UIB-PTR.
         IF UIBFCTR = NORESP
           MOVE UIBPCHAL   TO DLI-PCB-PTRS
           MOVE B-PCB1-PTR TO PCB1-PTR
         ELSE
           MOVE 'PSBU' TO ABEND-CODE
           PERFORM CICS-ABEND.
         SKIP2
     READ-TERMINAL.
         SKIP1
         EXEC CICS
            RECEIVE INTO(TRANS-DATA-IN)
             LENGTH(TDI-LENGTH)
         END-EXEC.
         SKIP2
     SEND-DATA.
         SKIP1
         EXEC CICS
             SEND FROM(TRANS-DATA-OUT)
             LENGTH(TDO-LENGTH)
         END-EXEC.
         SKIP3
     RELEASE-PSB.
         SKIP1
         MOVE TERM TO FUNCTION.
         CALL 'CBLTDLI' USING FUNCTION.
         EJECT
```

Figure E-4 (Part 4 of 7). CICS/VS Program Used by the Host System

```
READ-TERMINAL.
    SKIP1
    EXEC CICS
       RECEIVE INTO(TRANS-DATA-IN)
        LENGTH(TDI-LENGTH)
    END-EXEC.
    SKIP2
SEND-DATA.
    SKIP1
    EXEC CICS
        SEND FROM(TRANS-DATA-OUT)
        LENGTH(TDO-LENGTH)
    END-EXEC.
    SKIP3
RELEASE-PSB.
    SKIP1
    MOVE TERM TO FUNCTION.
    CALL 'CBLTDLI' USING FUNCTION.
    EJECT
CICS-ABEND.
    SKIP1
    EXEC CICS
        DUMP
        FROM(BLL-CELLS)
        LENGTH(32)
        DUMPCODE('ABLL')
    END-EXEC.
    SKIP1
    EXEC CICS
        DUMP
        FROM(DLIUIB)
        LENGTH(16)
        DUMPCODE('AUIB')
    END-EXEC.
    SKIP1
    EXEC CICS
        DUMP
        FROM(PCB-PTRS)
        LENGTH(48)
        DUMPCODE('APTR')
    END-EXEC.
    EJECT
    EXEC CICS
        DUMP
        FROM(PCB1)
        LENGTH(256)
        DUMPCODE('APCB')
    END-EXEC.
    SKIP3
    PERFORM RELEASE-PSB.
    SKIP1
    EXEC CICS
        ABEND
        ABCODE(ABEND-CODE)
    END-EXEC.
    EJECT
```

Figure E-4 (Part 5 of 7). CICS/VS Program Used by the Host System

```
************************************************************
*                                                          *
*    GET INVENTORY.                                        *
*      THIS ROUTINE ATTEMPTS TO READ THE INVENTORY ITEM    *
*      FROM THE DL/I DATABASE BASED ON THE ITEM NUMBER     *
*      SENT FROM THE REQUESTING AS/400 SYSTEM.  IF THE      *
*      ITEM IS NOT FOUND ON THE DATABASE THE ROUTINE       *
*      WHICH CALLS THIS ONE RETURNS AN ERROR MESSAGE TO    *
*      THE AS/400 SYSTEM.  OTHERWISE THE 'AMOUNTS ON HAND'  *
*      ARE READ FROM THE DATABASE AND RETURNED TO THE      *
*      AS/400 SYSTEM.                                       *
*                                                          *
************************************************************
     SKIP1
 GET-INVTY.
     SKIP1
     MOVE GU TO FUNCTION.
     SKIP1
     CALL 'CBLTDLI' USING FUNCTION, PCB1,
         SEGMENT-MSININ01-IN, INVTY-SEGMENT-SSA.
     SKIP1
     IF UIBFCTR NOT = NORESP
       MOVE 'ITYU' TO ABEND-CODE
       PERFORM CICS-ABEND.
     EJECT
************************************************************
*                                                          *
*    GET QUANTITIES ON HAND                                *
*      THIS ROUTINE GETS THE QUANTITIES ON HAND FROM UP    *
*      TO 4 REMOTE LOCATIONS AND RETURNS THE QUANTITIES    *
*      TO THE REQUESTING IBM AS/400 SYSTEM.  IF QUANTITIES  *
*      ARE NOT FOUND FOR SOME LOCATIONS, ZEROS ARE         *
*      RETURNED IN THEIR 'ON HAND' FIELDS.                 *
*                                                          *
************************************************************
     SKIP1
 GET-QTYS-ON-HAND.
     SKIP1
     PERFORM CLEAR-QTYS.
     SKIP1
     MOVE 0      TO DLI-SWITCH.
     MOVE PCBOK  TO PCB1-STATUS.
     MOVE NORESP TO UIBFCTR.
     SKIP3
     PERFORM GET-QTYS.                           NOTE LOCATION 1.
     IF UIBFCTR = NORESP
       IF QUANTITY-SEGMENT-FOUND
         MOVE MSLCHAND TO TDO-MSLCHAND1
         MOVE MSLCLOCD TO TDO-FILL1
       ELSE
         NEXT SENTENCE
     ELSE
       MOVE 1 TO DLI-SWITCH
       GO TO GET-QTYS-EXIT.
     SKIP3
     IF NOT END-OF-CHAIN
     PERFORM GET-QTYS
```

Figure E-4 (Part 6 of 7). CICS/VS Program Used by the Host System

```
              IF UIBFCTR = NORESP
                IF QUANTITY-SEGMENT-FOUND
                  MOVE MSLCHAND TO TDO-MSLCHAND2
                ELSE
                   NEXT SENTENCE
              ELSE
                MOVE 1 TO DLI-SWITCH
                GO TO GET-QTYS-EXIT.
              EJECT
              IF NOT END-OF-CHAIN
              PERFORM GET-QTYS
              IF UIBFCTR = NORESP
                IF QUANTITY-SEGMENT-FOUND
                  MOVE MSLCHAND TO TDO-MSLCHAND3
                ELSE
                  NEXT SENTENCE
              ELSE
                MOVE 1 TO DLI-SWITCH
                GO TO GET-QTYS-EXIT.
              SKIP3
              IF NOT END-OF-CHAIN
              PERFORM GET-QTYS
              IF UIBFCTR = NORESP
                IF QUANTITY-SEGMENT-FOUND
                  MOVE MSLCHAND TO TDO-MSLCHAND4
                ELSE
                  NEXT SENTENCE
              ELSE
                MOVE 1 TO DLI-SWITCH
                GO TO GET-QTYS-EXIT.
              SKIP3

MEMBER NE  ICII
       GET-QTYS-EXIT.  EXIT.
           EJECT
       CLEAR-QTYS.
           SKIP1
           MOVE ZEROS TO TDO-MSLCHAND1,
                         TDO-MSLCHAND2,
                         TDO-MSLCHAND3,
                         TDO-MSLCHAND4.
           SKIP3
       GET-QTYS.
           SKIP1
           MOVE GNP TO FUNCTION.
           SKIP1
           CALL 'CBLTDLI' USING FUNCTION, PCB1,
             SEGMENT-MSLCIN01-IN, QTYS-SEGMENT-SSA.
           SKIP3
       MOVE-QUANTITY.
           SKIP1
           IF MSLCLOCD = '10'
             MOVE MSLCHAND TO TDO-MSLCHAND1.
           SKIP1
           IF MSLCLOCD = '20'
             MOVE MSLCHAND TO TDO-MSLCHAND2.
           SKIP1
           IF MSLCLOCD = '30'
             MOVE MSLCHAND TO TDO-MSLCHAND3.
           SKIP1
           IF MSLCLOCD = '40'
             MOVE MSLCHAND TO TDO-MSLCHAND4
           ELSE
             NEXT SENTENCE.
```

Figure E-4 (Part 7 of 7). CICS/VS Program Used by the Host System

Example 2: AS/400 System to System/370 System (IMS/VS)

This example consists of an AS/400 ILE RPG/400 program with DDS keywords, talking to a System/370 COBOL IMS/VS program.

Not all programming considerations or techniques are illustrated in this example. You should review the example before you begin application design and coding.

ILE RPG/400 Program for the AS/400 System (Program A)

The following program (PROGARPG) is used on the AS/400 system.

When PROGARPG is called, a display is presented. This display presents a single inquiry line which is 23 bytes long. The operator enters the item number on the inquiry line, and the local database FILEA is searched. If a matching item is found, up to four quantities are displayed for that item.

If the item number is not in the local database, a request is built and sent to the host system. Program MSCGT005 is started and searches the host database; the matching item number and its associated quantities are returned to PROGARPG and are presented on the inquiry display.

DDS Sources: The DDS sources for FILEA, WSFILE, and RMFILE, are illustrated in Figure E-5, Figure E-6 on page E-18, and Figure E-7 on page E-18.

```
A*****************************************************************
A*                                                               *
A*     DDS SOURCE FOR THE MASTER FILE (FILEA)                    *
A*                                                               *
A*****************************************************************
A                                       LIFO
A          R MASTER
A            ITMNUM        23A
A            FILLER         3A
A            QTY1           6  0
A            QTY2           6  0
A            QTY3           6  0
A            QTY4           6  0
A          K ITMNUM
```

Figure E-5. DDS Source for File FILEA

```
A*****************************************************************
A*                                                               *
A*     DDS SOURCE FOR THE LOCAL DISPLAY FILE (WSFILE)            *
A*                                                               *
A*****************************************************************
A*
A                                       CA01
A                                       CA07
A          R WSFILEIN
A            ITMNUM        23   I  2 10
A  41                           O  4 10'INVALID ITEM NUMBER ENTERED'
A  42                           O  4 10'ITEM NUMBER NOT FOUND'
A                               O 12 10'PRESS CMD KEY 7 TO TERMINATE'
A          R WSFILEOT
A            ITMNUM        23   O  2 10
A            QTY1           6  0O  4 12
A            QTY2           6  0O  5 12
A            QTY3           6  0O  6 12
A            QTY4           6  0O  7 12
A                               O 12 10'PRESS CMD KEY 7 TO TERMINATE OR'
A                               O 13 10'CMD KEY 1 FOR ANOTHER INQUIRY'
```

Figure E-6. DDS Source for File WSFILE

```
A******************************************************************
A*                                                                *
A*     DDS FOR THE ICF COMMUNICATION FILE (RMFILE)                *
A*                                                                *
A******************************************************************
A          R RMDTCH
A  51                                   DETACH
A*
A          R RMEOS
A  52                                   EOS
A*
A          R RMDATA
A            ERRFLD         3A
A            FILLR1        29A
A            ITMNUM        23A
A            FILLR2       146A
A            QTY1           6  0
A            QTY2           6  0
A            QTY3           6  0
A            QTY4           6  0
A*
A          R RMEVOK
A                                       SECURITY(2 &PASSWD)
A  53                                   EVOKE(&PGMID &ITMNUM)
A  53                                   INVITE
A            PGMID          8A  P
A            PASSWD         4A  P
A            ITMNUM        23A  P
```

Figure E-7. DDS Source for File RMFILE

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

```
CRTICFF FILE(RMFILE)
        SRCFILE(QDDSSRC)
        SRCMBR(ICFFILE1)
        ACQPGMDEV(PGMDEV)
```

The command needed to define the program device entry is:

```
ADDICFDEVE FILE(RMFILE)
           PGMDEV(PGMDEV)
           RMTLOCNAME(IMSLOC)
           CMNTYPE(*SNUF)
           DEV(YDEV)
           HOST(*IMS) or (*DEVD)
```

ILE RPG/400 Program: Figure E-8 shows the ILE RPG/400 Program A for the AS/400 system.

```
F***************************************************************
F*                                                             *
F*     FILE DESCRIPTION SPECIFICATIONS                         *
F*                                                             *
F***************************************************************
FFILEA   IF  E           K        DISK
FWSFILE  CF  E                    WORKSTN
FRMFILE  CF  E                    WORKSTN
F                                              KNUM       1
F                                              KID    DEV
C***************************************************************
C*                                                             *
C*    RPG CALCULATION SPECIFICATIONS                           *
C*                                                             *
C***************************************************************
C*
C***************************************************************
C*                                                             *
C*    PROMPT FOR ITEM NUMBER OR DISPLAY ERROR MESSAGE          *
C*                                                             *
C***************************************************************
C*
C           ITMINQ    TAG
C                     EXFMTWSFILEIN               12
C   KG 12             GOTO EOJ
C                     SETOF                     41
C           *BLANK    COMP ITMNUM                   41
C  N41      *ZERO     COMP ITMNUM                   41
C   41                GOTO ITMINQ
C*
C*    SEARCH LOCAL MASTER FILE FOR ITEM
C*
C                     SETOF                     134142
C           ITMNUM    CHAINMASTER               1315
C   15                GOTO EOJ
C*
C*    DISPLAY RESULTS OF INQUIRY
C*
C           DSPLY     TAG
C  N13                EXFMTWSFILEOT
C  N13 KA             GOTO ITMINQ
C  N13 KG             GOTO EOJ
C*
```

Figure E-8 (Part 1 of 2). ILE RPG/400 Program A for the AS/400 System

```
C*    ITEM NOT ON LOCAL MASTER - SEND INQUIRY TO HOST
C*
C   13               SETOF                     13
C                    SETON                     14
C                    MOVEL'PGMDEV'  DEV
C                    MOVE 'MSCGT005'PGMID
C                    MOVE '0023'    PASSWD
C                    SETON                     53
C                    WRITERMEVOK                      EVOKE
C                    SETOF                     53
C                    READ RMDATA                   50
C          '***'     COMP ERRFLD                   42
C                    SETON                     51
C                    WRITERMDTCH                      DETACH
C                    SETOF                     51
C   42               GOTO ITMINQ
C                    GOTO DSPLY
C*
C*    CMD KEY 7 PRESSED - TERMINATE SESSION
C*
C          EOJ       TAG
C  N14               SETON                     52
C  N14               WRITERMEOS                       EOS
C                    SETON                     LR
```

Figure E-8 (Part 2 of 2). ILE RPG/400 Program A for the AS/400 System

IMS/VS Program Used by the Host System (Program B)

Figure E-9 is the IMS application program used by the host system to communicate with an AS/400 system.

```
 IDENTIFICATION DIVISION.
     SKIP3
*****************************************************************
*                                                               *
*    ICII       INVENTORY INQUIRY                               *
*               *MSCGT005*                                      *
*****************************************************************
     SKIP3
   PROGRAM-ID.  MSCGT005.
     SKIP1
     SKIP3
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
     SKIP1
*  SOURCE-COMPUTER.  IBM-370-168 WITH DEBUGGING MODE.
   SOURCE-COMPUTER.  IBM-370-168.
   OBJECT-COMPUTER.  IBM-370-168.
 INPUT-OUTPUT SECTION.
   FILE-CONTROL.
     SELECT PRNT-FILE ASSIGN TO UR-S-PRNTFILE.
     EJECT
 DATA DIVISION.
 FILE SECTION.
     SKIP1
 FD  PRNT-FILE LABEL RECORDS OMITTED, RECORD CONTAINS 132
     CHARACTERS.
 01  PRINTLINE.
     02  ITEM1           PIC X(80).
     02  FILLER          PIC X(12).
     02  ITEM2           PIC X(40).
     SKIP3
 WORKING-STORAGE SECTION.
     SKIP1
 77  I              PIC 99.
     SKIP3
*****************************************************************
*                                                               *
*    TRANSACTION DATA RECEIVED FROM IBM AS/400 SYSTEM           *
*                                                               *
*****************************************************************
     SKIP1
 01  TRANS-DATA-IN.
     05  TDI-FILL1           PIC X(1).
     05  TDI-MSINITEM        PIC X(23).
     SKIP3
*****************************************************************
*                                                               *
*    TRANSACTION DATA SENT BACK TO IBM AS/400 SYSTEM            *
*                                                               *
*****************************************************************
     SKIP1
```

Figure E-9 (Part 1 of 6). IMS/VS Program Used by the Host System

```
 01  TRANS-DATA-OUT.
     05  LL               PIC 999 COMP-4 VALUE 260.
     05  FILLER           PIC 999 COMP-4 VALUE ZERO.
*    OUTPUT DATA FOR APPLICATION RESPONSE
     05  TDO-FILL1           PIC X(32)       VALUE SPACES.
     05  TDO-MSINITEM        PIC X(23).
     05  FILLER              PIC X(5)        VALUE SPACES.
     05  TDO-DESC            PIC X(10).
     05  FILLER              PIC X(130)      VALUE SPACES.
     05  TDO-MGLCHAND-VALUE.
       10  TDO-MSLCHAND      PIC S9(6) OCCURS 4 TIMES.
     05  TDO-FILL3           PIC X(32)       VALUE SPACES.
     SKIP1
 01  ERROR-MESSAGE REDEFINES TRANS-DATA-OUT.
     05  LL               PIC 999 COMP-4.
     05  FILLER           PIC 999 COMP-4.
*    OUTPUT DATA FOR APPLICATION RESPONSE
     05  EM-MESSAGE      PIC X(51).
     05  EM-FILLER       PIC X(205).
     SKIP3
*    CONVERSATIONAL INPUT/OUTPUT DATA AREA FOR IO-PCB.
 01  SPA-AREAS.
     02  LL                  PIC 999 COMP-4, VALUE IS 270.
     02  FILL                PIC X(4).
     02  TRAN-CODE           PIC X(8).
     02  WORK-AREA.
         05  WORK-1          PIC X(24).
         05  WORK-2          PIC X(232).
 66  SPA-IN RENAMES FILL THROUGH WORK-1.
 66  SPA-OUT RENAMES FILL THRU WORK-2.
     EJECT
*****************************************************************
*                                                               *
*    DL/I INVENTORY DATABASE SEGMENT INPUT AREA                 *
*                                                               *
*****************************************************************
     SKIP1
 01  SEGMENT-MSININ01-IN.
     05  MSINITEM            PIC X(23).
     05  MSINBCOD            PIC X(2).
     05  MSINCOST            PIC S9(7)       COMP-3.
     05  MSINDESC            PIC X(10).
     05  FILLER              PIC X(6) VALUE SPACES.
     SKIP3
 01  SEGMENT-MSLCIN01-IN.
     05  MSLCLOCD            PIC 9(2).
     05  MSLCHAND            PIC S9(9)       COMP-3.
     05  MSLCALOC            PIC S9(9)       COMP-3.
     05  MSLCCLOC            PIC S9(9)       COMP-3.
     05  MSLCBORD            PIC S9(9)       COMP-3.
     05  MSLCWORK            PIC S9(9)       COMP-3.
     05  FILLER              PIC X(8) VALUE SPACES.
     SKIP3
```

Figure E-9 (Part 2 of 6). IMS/VS Program Used by the Host System

```
******************************************************************
*                                                                *
*    DL/I INVTY DATABASE SEGMENT SEARCH ARGUMENTS (SSA'S)        *
*                                                                *
******************************************************************
     SKIP1
 01  INVTY-SEGMENT-SSA.
     05  FILLER          PIC X(19)   VALUE 'MSININ01(MSINITEM ='.
     05  SSA-MSINITEM    PIC X(23).
     05  FILLER          PIC X(1)    VALUE ')'.
     SKIP3
 01  QTYS-SEGMENT-SSA.
     05  FILLER          PIC X(9)    VALUE 'MSLCIN01'.
     SKIP3
******************************************************************
*                                                                *
*    MISCELLANEOUS WORKING-STORAGE AREAS                         *
*                                                                *
******************************************************************
     SKIP1
 01  DLI-FUNCTIONS.
     05  GU              PIC X(4)    VALUE 'GU  '.
     05  GN              PIC X(4)    VALUE 'GN  '.
     05  GNP             PIC X(4)    VALUE 'GNP '.
     05  ISRT            PIC X(4)    VALUE 'ISRT'.
     SKIP3
 01  MSG-NOT-FOUND.
     05  FILLER          PIC X(11)   VALUE '***ERROR***'.
     05  FILLER          PIC X(6)    VALUE ' ITEM '.
     05  MSG-MSINITEM    PIC X(23).
     05  FILLER          PIC X(12)   VALUE ' NOT FOUND. '.
     SKIP3
     EJECT
 LINKAGE SECTION.
     SKIP1
 01  IO-PCB.
     05  TERM-NAME       PIC X(8).
     05  FILLER          PIC XX.
     05  IO-STATUS       PIC XX.
     05  IO-PREFIX       PIC X(12).
     05  FILLER          PIC X(8).
 01  PCB1.
     05  PCB1-DBD-NAME   PIC X(8).
     05  PCB1-SEG-LEVEL  PIC X(2).
     05  PCB1-STATUS     PIC X(2).
       88  SEGMENT-FOUND              VALUE '  '.
       88  SEGMENT-END                VALUE 'GE'.
     05  PCB1-PROC-OPTN  PIC X(4).
     05  PCB1-FILL1      PIC S9(5)    COMP.
     05  PCB1-SEG-NAME   PIC X(8).
     05  PCB1-LENG-KFBA  PIC S9(5)    COMP.
     05  PCB1-NUMBOSSEG  PIC S9(5)    COMP.
     05  PCB1-KEY-FBA    PIC X(256).
     EJECT
```

Figure E-9 (Part 3 of 6). IMS/VS Program Used by the Host System

```
******************************************************************
*                                                                *
*    MAIN LINE ROUTINE                                           *
*                                                                *
******************************************************************
     SKIP1
 PROCEDURE DIVISION.
     SKIP1
 DECLARATIVES.
 DEBUG-IMS SECTION.  USE FOR DEBUGGING ON ALL REFERENCES
     PCB1.
 DEBUG-IMS-RTN.
D    EXHIBIT NAMED PCB1.
 END DECLARATIVES.
     SKIP3
 BEGIN-IMS SECTION.
 BEGIN-PROCESSING.
     ENTRY 'DLITCBL' USING     IO-PCB, PCB1.
     OPEN OUTPUT PRNT-FILE.
     READY TRACE.
 READ-TERMINAL.
     CALL 'CBLTDLI' USING GU, IO-PCB, SPA-IN.
D    EXHIBIT NAMED IO-PCB, SPA-IN.
     IF IO-STATUS EQUAL TO 'QC' GO TO CLOSING,
     ELSE IF IO-STATUS NOT EQUAL SPACES MOVE IO-PCB TO ITEM1,
          MOVE 'GU FOR ID-PCB' TO ITEM2, GO TO ABEND.
*    CALL 'CBLTDLI' USING GN, IO-PCB, SPA-IN.
D    EXHIBIT NAMED IO-PCB, SPA-IN.
     MOVE 'GN FOR IO-PCB, TO ITEM2.
*
     IF IO-STATUS EQUAL 'QD' NEXT SENTENCE,
     ELSE IF IO-STATUS NOT = SPACES MOVE IO-PCB TO ITEM1,
     GO TO ABEND.
     MOVE WORK-1 TO TRANS-DATA-IN.
*
     MOVE TDI-MSINITEM TO SSA-MSINITEM.
     SKIP3
********************************************************************
*                                                                  *
*    GET INVENTORY                                                 *
*      THIS ROUTINE ATTEMPTS TO READ THE INVENTORY ITEM FROM THE   *
*      DL/I DATABASE BASED ON THE ITEM NUMBER SENT FROM THE        *
*      REQUESTING AS/400 SYSTEM.  IF THE ITEM IS NOT FOUND ON THE   *
*      DATABASE, THE ROUTINE WHICH CALLS THIS ONE RETURNS AN       *
*      ERROR MESSAGE TO AS/400 SYSTEM.  OTHERWISE THE 'AMOUNTS OF   *
*      HAND' ARE READ FROM THE DATABASE AND RETURNED TO            *
*      THE AS/400 SYSTEM.                                           *
*                                                                  *
********************************************************************
     SKIP1
 GET-INVTY.
     CALL 'CBLTDLI' USING GU, PCB1, SEGMENT-MSININ01-IN,
                    INVTY-SEGMENT-SSA.
```

Figure E-9 (Part 4 of 6). IMS/VS Program Used by the Host System

```
D    EXHIBIT NAMED SEGMENT-MSININ01-IN.
     IF SEGMENT-FOUND
          MOVE ZEROS TO TDO-MSLCHAND-VALUE,
          MOVE TDI-MSINITEM TO TDO-MSINITEM,
          MOVE MSINDESC TO TDO-DESC,
          PERFORM GET-QTYS UNTIL SEGMENT-END,
          PERFORM SEND-DATA,
      SKIP2
******************************************************************
*                                                                *
*    THIS ROUTINE IS EXECUTED IF THE INQUIRY INVENTORY           *
*    ITEM READ IS NOT FOUND IN DATABASE.  A MESSAGE              *
*    IS BUILT TO BE SENT BACK TO THE ORIGINATING LOCATION.       *
*    MESSAGE IS:                                                 *
*                                                                *
*    ***ERROR*** ITEM #########X(23)######### NOT FOUND          *
*                                                                *
******************************************************************
     ELSE IF SEGMENT-END
            MOVE TDI-MSINITEM TO MSG-MSINITEM,
            MOVE SPACES TO EM-FILLER,
            MOVE MSG-NOT-FOUND TO EM-MESSAGE,
            PERFORM SEND-DATA,
     SKIP2
******************************************************************
*                                                                *
*    FORCE ABEND, ERROR IN PROGRAM, AND DEBUG                    *
*                                                                *
******************************************************************
     SKIP1
          ELSE MOVE PCB1 TO ITEM1, MOVE 'GU ON PCB1' TO ITEM2,
             GO TO ABEND.
 CLOSING.  CLOSE PRNT-FILE.
*    SET TRANSACTION CODE IN SPA TO TERMINATE CONVERSATION
     MOVE SPACES TO TRAN-CODE.
     CALL 'CBLTDLI' USING ISRT, IO-PCB, SPA-AREAS.
*
         RESET TRACE.
     STOP RUN.
******************************************************************
*                                                                *
*    CLOSED SUBROUTINES                                          *
*                                                                *
******************************************************************
     SKIP1
     EJECT
******************************************************************
*                                                                *
*    GET QUANTITIES ON HAND                                      *
*      THIS ROUTINE GETS THE QUANTITIES ON HAND FROM UP TO 4     *
*      REMOTE LOCATIONS AND RETURNS THE QUANTITIES TO THE        *
*      REQUESTING IBM AS/400 SYSTEM.  IF  QUANTITIES ARE NOT     *
*      FOUND FOR SOME LOCATIONS, ZEROS ARE RETURNED IN THEIR     *
*      'ON HAND' FIELDS.                                         *
*                                                                *
     SKIP1
```

Figure E-9 (Part 5 of 6). IMS/VS Program Used by the Host System

```
 GET-QTYS.
     CALL 'CBLTDLI' USING GNP, PCB1, SEGMENT-MSLCI01-IN,
                      QTYS-SEGMENT-SSA.
D    EXHIBIT NAMED SEGMENT-MSLCIN01-IN.
     IF SEGMENT-FOUND    PERFORM DETERMINE-LOCATION,
                      MOVE MSLCHAND TO TDO-MSLCHAND (I),
     ELSE IF NOT SEGMENT-END MOVE PCB1 TO ITEM1, MOVE
         'GNP ON PCB1' TO ITEM2,  GO TO ABEND.
     SKIP3
 DETERMINE-LOCATION.
     IF MSLCLOCD - 21 MOVE 1 TO I,
     ELSE IF MSLCLOCD = 41 MOVE 2 TO I,
     ELSE IF MSLCLOCD = 51 MOVE 3 TO I,
     ELSE IF MSLCLOCD = 81 MOVE 4 TO I.
     SKIP3
*****************************************************************
*                                                               *
*    SEND IMS MESSAGE TO QUEUE FOR ROUTING TO ORIGINATING       *
*    AS/400 LOCATION                                            *
*                                                               *
*****************************************************************
     SKIP1
 SEND-DATA.
*    SET TRANSACTION CODE IN SPA TO END CONVERSATION
     MOVE SPACES TO TRAN-CODE.
*    CALL 'CBLTDLI' USING ISRT, IO-PCB, SPA-AREAS.
*
*    INSERT DATA MESSAGE FOR TRANSMISSION TO REQUESTER
*
     CALL 'CBLTDLI' USING ISRT, IO-PCB, TRANS-DATA-OUT.
D    EXHIBIT NAMED TRANS-DATA-OUT.
     IF IO-STATUS NOT EQUAL SPACES MOVE TO-PCB TO ITEM1,
        MOVE 'ISRT FOR IO-PCB' TO ITEM2,  GO TO ABEND.
     SKIP3
 ABEND.
     WRITE PRINTLINE AFTER ADVANCING 2 LINES.
     MOVE SPACES TO PRINTLINE.  MOVE 'ABEND OCCURRED' TO ITEM2.
     WRITE PRINTLINE.  GO TO CLOSING.
```

Figure E-9 (Part 6 of 6). IMS/VS Program Used by the Host System

Example 3: AS/400 System to System/370 System (CICS/VS)

The following example consists of an AS/400 ILE C/400 program using system-supplied ICF formats, communicating to a System/370 COBOL CICS/VS program. For the program used by the host system to communicate with the AS/400 system, see “CICS/VS Program Used by the Host System (Program B)” on page E-10.

As discussed in previous examples, not all programming considerations or techniques are illustrated in this example. You should review the example before you begin to design and code your application.

ILE C/400 Program for the AS/400 System (Program A)

The following program is used on the AS/400 system.

When the PROGAC program is called, a display is presented. This display presents a single inquiry line which is 23 bytes long. The operator enters the item number on the inquiry line, and the local database file, FILEA, is searched. If a matching item is found, up to four quantities are displayed for that item.

If an item number is not in the local database, a request is built and sent to the host system. The ICII program is started and searches the host system database. The matching item number and its quantities are returned to the PROGAC program and are presented on the inquiry display.

DDS Sources: The DDS sources for DISPFILE and FILEA are illustrated in Figure E-10 and Figure E-11 on page E-28.

```
A****************************************************************
A*                                                              *
A*     LOCAL DISPLAY FILE                                       *
A*                                                              *
A****************************************************************
A*
A                                       CA01
A                                       CA07(99)
A          R WSFILEIN
A            ITMNUM        23   I  2 10
A  01                           O  4 10'INVALID ITEM NUMBER ENTERED'
A  02                           O  4 10'ITEM NUMBER NOT FOUND'
A                               O 12 10'PRESS CMD KEY 7 TO TERMINATE'
A          R WSFILEOT
A            ITMNUM        23   O  2 10
A            QTY1           6  00  4 12
A            QTY2           6  00  5 12
A            QTY3           6  00  6 12
A            QTY4           6  00  7 12
A            MSG           80   O  8 10
A            RETCOD         4   O 10 10
A            REASON        30   O 11 10
A            FILLER        95   O 12 10
A                               O 14 10'PRESS CMD KEY 7 TO TERMINATE OR'
A                               O 15 10'CMD KEY 1 FOR ANOTHER INQUIRY'
```

Figure E-10. DDS Source for DISPFILE

```
A*****************************************************************
A*                                                               *
A*     DDS SOURCE FOR THE MASTER FILE (FILEA)                    *
A*                                                               *
A*****************************************************************
A                                       LIFO
A          R MASTER
A            ITMNUM        23A
A            FILLER         3A
A            QTY1           6  0
A            QTY2           6  0
A            QTY3           6  0
A            QTY4           6  0
A          K ITMNUM
```

Figure E-11. DDS Source for File FILEA

Program Device Entry Definition: The command needed to define the program device entry is:

```
ADDICFDEVE FILE(*LIBL/QICDMF)
           PGMDEV(SNUFDEVICE)
           RMTLOCNAME(CICSLOC)
           CMNTYPE(*SNUF)
           DEV(XDEV)
           APPID(CICS)
           HOST(*DEVD) or (*CICS)
```

ILE C/400 Program: Figure E-12 on page E-29 shows the ILE C/400 program PROGAC for the AS/400 system.

```
/**************************************************/
/*   PROGAC  ITEM INQUIRY WRITTEN IN ILE-C        */
/**************************************************/
#pragma mapinc("dspf","icflib/dispfile(*all)","both indicators","p z")
#include "dspf"
#pragma mapinc("fileainc", "icflib/filea(*all)", "input", "p z")
#include "fileainc"
#include <stdio.h>                 /* Standard I/O header           */
#include <recio.h>                 /* Record I/O header             */
#include <stdlib.h>                /* General utilities             */
#include <stddef.h>                /* Standard definitions          */
#include <string.h>                /* String handling utilities     */
#include <xxfdbk.h>                /* Feedback area structures      */

#define ERROR 1                    /* error occured                 */
#define NOERROR 0                  /* no error occured              */
#define ION  '1'                   /* indicator set on              */
#define IOFF '0'                   /* indicator set off             */
#define SPACE ' '                  /* spaces for memset             */
#define BLNK23 "                       " /* 23 blanks               */
#define ZERO23 "00000000000000000000000" /* 23 zeros                */
/********************************************************************/
/* Global Variables                                                 */
/********************************************************************/
_RFILE *icfptr;                    /* ptr to ICF file               */
_RFILE *dspfptr;                   /* ptr to display file           */
_RFILE *fileaptr;                  /* ptr to file A                 */
_RIOFB_T *rio_fbk;                 /* ptr to partial I/O feedback   */
_XXIOFB_T *comm_fdbk;              /* ptr to common I/O feedback    */
_XXIOFB_DSP_ICF_T *icf_fdbk;       /* ptr to icf feedback           */
/********************************************************************/
/* Structure to be used for return code                             */
/********************************************************************/
struct {
    char major??(2??);
    char minor??(2??);
    }return_code;
/********************************************************************/
/* Structure to be used in the evoke processing                     */
/********************************************************************/
struct {
    char proc_name??(8??);
    char password??(8??);
    char user_id??(8??);
    char libr_name??(8??);
    char filler??(20??);
    char data_length??(4??);
    char evok_data??(23??);
    }evok_record;
struct {
    char data??(32??);
    char number_in??(23??);
    char filler2??(145??);
    char icf_qty1??(6??);
    char icf_qty2??(6??);
    char icf_qty3??(6??);
    char icf_qty4??(6??);
    char filler3??(32??);
    }icf_record;
/********************************************************************/
/* Structure to be used in the end of transaction processing        */
/********************************************************************/
struct {
    char length??(4??);
    char data??(80??);
    }eot_rec;
 char input_buffer??(4096??);
```

Figure E-12 (Part 1 of 6). ILE C/400 Program for the AS/400 System

```
/********************************************************************/
/* Structure to be used to write to display file                    */
/********************************************************************/
ICFLIB_DISPFILE_WSFILEIN_o_t  wsfilein_o; /* display file input      */
ICFLIB_DISPFILE_WSFILEIN_i_t  wsfilein_i; /* display file input      */
ICFLIB_DISPFILE_WSFILEOT_o_t  wsfileot_o; /* display file output     */
ICFLIB_DISPFILE_WSFILEOT_i_t  wsfileot_i; /* display file output     */
ICFLIB_FILEA_MASTER_i_t  itmfile;         /* data file structure     */
/********************************************************************/
/* Prototyping of routines                                          */
/********************************************************************/
int open_files(void);
int read_filea(void);
int icf(void);
int logon_evoke(void);
void send_eos(void);
void close_files(void);
int check_return_code(void);
/********************************************************************/
/* main procedure begin                                             */
/********************************************************************/
main()
 {
   if (open_files() == ERROR)
      exit(ERROR);
    /****************************************************************/
    /* Display the screen that will request the number.             */
    /* If CMD7(99) then close the files and end.                    */
    /* Set up indicators to display errors if item number           */
    /* is zero or spaces.                                           */
    /****************************************************************/
    wsfilein_i.IN99ﬄ0“ = IOFF;
    wsfileot_i.IN99ﬄ0“ = IOFF;
    while ((wsfilein_i.IN99ﬄ0“ == IOFF) & (wsfileot_i.IN99ﬄ0“ == IOFF))
      {
      _Rformat(dspfptr, "WSFILEIN");
      _Rwrite(dspfptr, &wsfilein_o, sizeof(wsfilein_o));
      _Rreadn(dspfptr, &wsfilein_i, sizeof(wsfilein_i), __DFT);
      if (wsfilein_i.IN99ﬄ0“ == IOFF)
        {
        wsfilein_o.IN01ﬄ0“ = IOFF;
        wsfilein_o.IN02ﬄ0“ = IOFF;
        memset(wsfileot_o.RETCOD, SPACE , 4);
        memset(wsfileot_o.REASON, SPACE , 30);
        if ((strcmp(wsfilein_i.ITMNUM, BLNK23) == 0) ||
            (strcmp(wsfilein_i.ITMNUM, ZERO23) == 0))
           wsfilein_o.IN01ﬄ0“ = ION;
        else
          {
          if (read_filea() == ERROR)
             if (icf() == ERROR)
               wsfilein_o.IN02ﬄ0“ = ION;
          }                         /* if not 0 or " "              */
        if (wsfilein_o.IN02ﬄ0“ == IOFF & wsfilein_o.IN01ﬄ0“ == IOFF)
          {
          _Rformat(dspfptr, "WSFILEOT");
          _Rwrite(dspfptr, &wsfileot_o, sizeof(wsfileot_o));
          _Rreadn(dspfptr, &wsfileot_i, sizeof(wsfileot_i), __DFT);
          }                         /* end send data out to display  */
        }                   /* end if CMD7 was issued in Display file*/
      }                             /* end while CMD7 not issued     */
      close_files();
 }                                  /* end main procedure            */
```

Figure E-12 (Part 2 of 6). ILE C/400 Program for the AS/400 System

```
/********************************************************************/
/* open files procedure                                             */
/********************************************************************/
int
open_files(void)
 {
    /****************************************************************/
    /* open ICF file                                                */
    /****************************************************************/
   if ((icfptr = _Ropen("QICDMF", "ar+ indicators=y riofb=y")) == NULL)
      return(ERROR);
    /****************************************************************/
    /* open file A                                                  */
    /****************************************************************/
   if ((fileaptr = _Ropen("FILEA", "rr+ riofb=y")) == NULL)
      {
      _Rclose(icfptr);
      return(ERROR);
      }
    /****************************************************************/
    /* open display file                                            */
    /****************************************************************/
   if ((dspfptr = _Ropen("dispfile", "ar+ riofb=y"))
                                        == NULL)
      {
      _Rclose(icfptr);
      _Rclose(fileaptr);
      return(ERROR);
      }
   return(NOERROR);
 }                                /* end open files procedure      */
/********************************************************************/
/* read filea procedure                                             */
/********************************************************************/
int
read_filea(void)
 {
    _Rformat(fileaptr, "ITEM");
    rio_fbk = _Rreadk(fileaptr, &itmfile, sizeof(itmfile), __KEY_EQ,
                     wsfilein_i.ITMNUM, sizeof(itmfile.ITMNUM));
    if (rio_fbk->num_bytes == 0)
       return(ERROR);
    strncpy(wsfileot_o.ITMNUM, itmfile.ITMNUM, 23);
    strncpy(wsfileot_o.QTY1, itmfile.QTY1, 6);
    strncpy(wsfileot_o.QTY2, itmfile.QTY2, 6);
    strncpy(wsfileot_o.QTY3, itmfile.QTY3, 6);
    strncpy(wsfileot_o.QTY4, itmfile.QTY4, 6);
    return(NOERROR);
 }                                /* end of read filea procedure   */
```

Figure E-12 (Part 3 of 6). ILE C/400 Program for the AS/400 System

```
/*********************************************************************/
/* ICF file procedure                                                */
/*********************************************************************/
int
icf(void)
 {
    /*****************************************************************/
    /* acquire SNUF session                                          */
    /*****************************************************************/
   _Racquire(icfptr, "SNUFDEVICE");
   if (check_return_code() == NOERROR)
     {
    /*****************************************************************/
    /* evoke processing evoke ICII at host                           */
    /*****************************************************************/
     if (logon_evoke() == NOERROR)
       {
    /*****************************************************************/
    /* read host data and end transaction                            */
    /*****************************************************************/
       _Rreadindv(icfptr, &icf_record, sizeof(icf_record), __DFT);
       if (check_return_code() == NOERROR)
         {
           _Rformat(icfptr, "$$SENDET");
           strncpy(eot_rec.length, "0080", 4);
           _Rwrite(icfptr, &eot_rec, sizeof(eot_rec));
           if (check_return_code() == NOERROR)
             {
    /*****************************************************************/
    /* check if host data is valid                                   */
    /*****************************************************************/
             if (strncmp(icf_record.data, "***", 3) == 0)
               return(ERROR);
             else
               {
               strncpy(wsfileot_o.ITMNUM, icf_record.number_in, 23);
               strncpy(wsfileot_o.QTY1, icf_record.icf_qty1, 6);
               strncpy(wsfileot_o.QTY2, icf_record.icf_qty2, 6);
               strncpy(wsfileot_o.QTY3, icf_record.icf_qty3, 6);
               strncpy(wsfileot_o.QTY4, icf_record.icf_qty4, 6);
               return(NOERROR);
               }                    /* if data = ***                 */
             }                      /* end if end transaction was ok */
         }                          /* read from ICF was good        */
       }                            /* end logon and evoke ICII      */
    /*****************************************************************/
    /* if there was an error in the evoke processing, ICF read       */
    /* processing or the end of transaction processing then the      */
    /* session should be taken down.                                 */
    /*****************************************************************/
       send_eos();
     }                              /* end if acquire was good       */
    return(NOERROR);
 }                                  /* end of process ICF            */
```

Figure E-12 (Part 4 of 6). ILE C/400 Program for the AS/400 System

```
/*******************************************************************/
/* Logon to CICS procedure                                         */
/*******************************************************************/
int logon_evoke(void)
 {
 /*************************************************************/
 /* set up evoke structure to logon to CICS using CSSN        */
 /*************************************************************/
 memset(evok_record.evok_data, SPACE, 24);
 strncpy(evok_record.proc_name, "CSSN    ", 8);
 strncpy(evok_record.password,"USER    ", 8);
 strncpy(evok_record.user_id, "USID    ", 8);
 strncpy(evok_record.data_length, "0024", 4);
 strncpy(evok_record.evok_data, "NAME=USER    PS=USID    ", 24);
 _Rformat(icfptr, "$$EVOK");
 _Rwrite(icfptr, &evok_record, sizeof(evok_record));
 if (check_return_code() == NOERROR)
   {
   /*************************************************************/
   /* read logon response from host                             */
   /*************************************************************/
   _Rreadindv(icfptr, &input_buffer, sizeof(input_buffer), __DFT);
   if (check_return_code() == NOERROR)
     {
     /*************************************************************/
     /* read detach response from host                            */
     /*************************************************************/
     _Rformat(icfptr, "DFTRCD");
     _Rreadn(icfptr, &input_buffer, sizeof(input_buffer), __DFT);
     if (check_return_code() == NOERROR)
       {
       /*************************************************************/
       /* set up evoke structure to start up ICII program           */
       /*************************************************************/
       memset(evok_record.evok_data, SPACE, 24);
       strncpy(evok_record.proc_name, "ICII    ", 8);
       strncpy(evok_record.data_length, "0023", 4);
       strncpy(evok_record.user_id, "USER    ", 8);
       strncpy(evok_record.password,"USID    ", 8);
       strncpy(evok_record.evok_data, wsfilein_i.ITMNUM, 23);
       _Rformat(icfptr, "$$EVOK");
       _Rwrite(icfptr, &evok_record, sizeof(evok_record));
       if (check_return_code() == NOERROR)
         {
         return(NOERROR);
         }                          /* end if data evoke is good     */
       }                            /* end if read detach indication */
     }                              /* end read LOGON response       */
   }                                /* end write LOGON to CICS evoke */
   return(ERROR);
 }                                  /* end of send end of session    */
/*******************************************************************/
/* send end of session procedure                                   */
/*******************************************************************/
void send_eos(void)
 {
   _Rformat(icfptr, "$$EOS");
   _Rwrite(icfptr, &icf_record, sizeof(icf_record));
 }                                  /* end of send end of session    */
/*******************************************************************/
/* close files procedure                                           */
/*******************************************************************/
void close_files(void)
 {
   _Rclose(icfptr);
   _Rclose(dspfptr);
   _Rclose(fileaptr);
 }                                  /* end of close files            */
```

Figure E-12 (Part 5 of 6). ILE C/400 Program for the AS/400 System

```
/********************************************************************/
/* Check return code procedure                                      */
/********************************************************************/
int
check_return_code(void)
 {
   comm_fdbk = _Riofbk(icfptr);
   icf_fdbk = (_XXIOFB_DSP_ICF_T *)((char *)comm_fdbk +
                   comm_fdbk->file_dep_fb_offset);
   if ((strncmp(icf_fdbk->major_ret_code, "00", 2) == 0) ||
       (strncmp(icf_fdbk->major_ret_code, "02", 2) == 0) ||
       (strncmp(icf_fdbk->major_ret_code, "03", 2) == 0))
     return(NOERROR);
   else                             /* error case                   */
     {
     strncpy(return_code.major, icf_fdbk->major_ret_code, 2);
     strncpy(return_code.minor, icf_fdbk->minor_ret_code, 2);
     memcpy(wsfileot_o.RETCOD, &return_code, 4);
     if (strncmp(icf_fdbk->major_ret_code, "04", 2) == 0)
    (strncpy(wsfileot_o.REASON, "Output Exception              ", 30));
     else
     if (strncmp(icf_fdbk->major_ret_code, "82", 2) == 0)
    (strncpy(wsfileot_o.REASON, "Unable to Acquire             ", 30));
     return(ERROR);
     }                              /* end if return is not good    */
 }                                  /* end of return code check     */
```

Figure E-12 (Part 6 of 6). ILE C/400 Program for the AS/400 System

Bibliography

The IBM publications listed here contain information you may need when you use AS/400 SNUF support. The following books are listed with their full title and order number.

- *ISDN Support*, SC41-5403.

  Contains information on connecting an AS/400 system to an integrated services digital network (ISDN) using the AS/400 integrated communications adapter.

- *ICF Programming*, SC41-5442.

  Supplies the application programmer with information needed to write communications programs that use the intersystem communications function (ICF) file. It also contains examples of communications programs and describes return codes.

- *Communications Management*, SC41-5406.

  Contains information on working with communications status, communications-related work management topics, communications errors, performance, aggregate line speed, and subsystem storage.

- *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.

- *Retail Communications Programming*, SC41-5448.

  Provides information on retail pass-through support.

- *3270 Device Emulation Support*, SC41-5408.

  Provides information for using the OS/400* binary synchronous communications (BSC) and 3270 device emulation for System Network Architecture (SNA).

- *DDS Reference*, SC41-5712.

  Contains information about coding data description specifications for files.

- *Languages: Systems Application Architecture* AD/Cycle* COBOL/400* User's Guide*, SC09-1383.

  Provides the information needed to write, test, and maintain ILE COBOL/400 programs for the AS/400 system.

- *Languages: Systems Application Architecture* AD/Cycle* RPG/400* User's Guide*, SC09-1348.

  provides the information needed to use the ILE RPG/400 programming language to code programs for the AS/400 system.

- *Languages: Systems Application Architecture* C/400* User's Guide*, SC09-1347.

  Provides the information needed to use the ILE C/400 programming language to code programs for the AS/400 system.

- *System/36 Environment Programming*, SC41-4730.

  Identifies the differences in the applications process in the System/36 environment on the AS/400 system.

- *CL Reference*, SC41-5722.

  Contains the commands, command parameters and syntax for the commands used in this guide.

- *System/38 Environment Programming*, SC41-3735.

  Describes the differences in the applications process in the System/38 environment on the AS/400 system.

- *Work Management*, SC41-5306.

  Provides information on how to create and change a work management environment.

- *Security – Reference*, SC41-5302.

  Contains information for the AS/400 system security officer about planning for security and setting up security on the system.

- *System Operation*, SC41-4203.

  Provides information on how to use the system unit operator display.

The following guides contain additional information you may need when you use this guide:

- *ACF/VTAM* Programming Guide*, SC23-0115.
- *CICS/OS/VS Messages and Codes*, SC33-0226.
- *CICS/OS/VS Version 1 Release 7 CICS-Supplied Transactions*, SC33-0240.
- *IBM Systems Network Architecture Format and Protocol Reference Manual: Architectural Logic*, SC30-3112.
- *IBM System/38 Data Communications Programmer's Guide*, SC21-7825.
- *IBM System/38 3270 Emulation Reference Manual and User's Guide*, SC21-7961.
- *IBM 3270 Information Display System, Data Stream Programmer's Reference*, GA23-0059.
- *IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide*, GA23-0061.
- *IBM 3270 SNA Programming Interface User's Guide*, SC21-9785.

- *IBM 5250 Information Display System Functions Reference Manual*, SA21-9247.
- *IMS/VS Version 1 Message Format Service User's Guide*, SH20-9053.
- *IMS/VS Version 2 Messages and Codes Reference Manual*, SC26-4174.
- *IMS/VS Version 2 for Remote SNA Systems Programming Guide*, SC26-4186.
- *Network Control Program System Support Programs and Emulation Program Resource Definition Reference*, SC30-3448.
- *Systems Network Architecture Formats*, GA27-3136.

Index

Reader Comments—We'd Like to Hear from You!

AS/400 Advanced Series
SNA Upline Facility Programming
Version 4

Publication No. SC41-5446-00

Overall, how would you rate this manual?

	Very Satisfied	Satisfied	Dissatis-fied	Very Dissatis-fied
Overall satisfaction				

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				
T H A N K Y O U !				

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.	

Reader Comments—We'd Like to Hear from You!
SC41-5446-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 542 IDCLERK
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-9986

Fold and Tape **Please do not staple** Fold and Tape

SC41-5446-00

Cut or Fold
Along Line

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC41-5446-00

Spine information:

AS/400 Advanced Series   SNA Upline Facility Programming   *Version 4*