# INTERCOMM

## PAGE FACILITY

ISOGON CORPORATION
330 Seventh Avenue, New York, New York 10001

**LICENSE: INTERCOMM TELEPROCESSING MONITOR**

## Publishing History

| Publication | Date | Remarks |
|---|---|---|
| First Edition | November 1973 | Introducing the feature. This manual corresponds to Intercomm Release 6.0. |
| Second Edition | January 1975 | General revisions and updates. |
| Third Edition | December 1981 | General revisions and updates. This edition corresponds to Intercomm Release 8.0. |
| Fourth Edition | March 1992 | General revisions and updates. This edition corresponds to Intercomm Releases 9 and 10. |

# PREFACE

Intercomm is a state-of-the-art teleprocessing monitor, executing on the IBM System/370 and System/390 family of computers and operating under the control of IBM Operating Systems (MVS/370, XA, and ESA). Intercomm monitors the transmission of messages to and from terminals, concurrent message processing, centralized access to I/O files, and the routine utility operations of editing input messages and formatting output messages, as required.

This manual documents the Intercomm Page Facility, which provides for the creation and subsequent access (browsing) of multiscreen collections of data for display on CRT terminals.

It is assumed that the reader is familiar with Intercomm. This manual is to be used in conjunction with the following:

- COBOL Programmers Guide
- PL/1 Programmers Guide
- Assembler Language Programmers Guide
- Operating Reference Manual
- Basic System Macros

INTERCOMM PUBLICATIONS

GENERAL INFORMATION MANUALS

- Concepts and Facilities
- Planning Guide

APPLICATION PROGRAMMERS MANUALS

- Assembler Language Programmers Guide
- COBOL Programmers Guide
- PL/1 Programmers Guide

SYSTEM PROGRAMMERS MANUALS

- Basic System Macros
- BTAM Terminal Support Guide
- Installation Guide
- Messages and Codes
- Operating Reference Manual
- System Control Commands

CUSTOMER INFORMATION MANUALS

- Customer Education Course Catalog
- Technical Information Bulletins
- User Contributed Program Description

FEATURE IMPLEMENTATION MANUALS

- Autogen Facility
- ASMF Users Guide
- DBMS Users Guide
- Data Entry Installation Guide
- Data Entry Terminal Operators Guide
- Dynamic Data Queuing Facility
- Dynamic File Allocation
- Extended Security System
- File Recovery Users Guide
- Generalized Front End Facility
- Message Mapping Utilities
- Model System Generator
- Multiregion Support Facility
- Page Facility
- Store/Fetch Facility
- SNA Terminal Support Guide
- TCAM Support Users Guide
- Utilities Users Guide

EXTERNAL FEATURES MANUALS

- SNA LU6.2 Support Guide

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

# Chapter 1

# INTRODUCTION

## 1.1 OVERVIEW

The Page Facility of Intercomm enables an application program (called a subsystem in Intercomm documentation) to create a response to one incoming message consisting of several pages of output messages which are destined, in general, for visual display terminals (CRTs). The Page Facility, in turn, allows the terminal operator to control transmission flow or display of those messages. The Page Facility accomplishes this by saving any messages passed to it by an application subsystem. Each message to be saved, or paged, is written to a BDAM data set called the Page Data Set. The terminal operator can then request a display of any page, or browse through these application-generated messages at his convenience. Furthermore, the Page Facility provides the operator with the capability of saving a group of pages, called a response, for later reference.

The Page Facility relieves the application programmer from developing logic to converse with the terminal operator to determine which messages to send. Instead, the application program can generate an entire set of output messages while the operator views only the individual messages desired by entering one or more control commands.

An application subsystem, for example, may be created to perform phonetic file retrievals by customer name. This type of retrieval may generate a long list of customer names with additional identifying information. If the output were destined for a hard copy terminal, the complete listing would be readily available. However, when several lines of output are directed to a visual display terminal, the viewing screen is limited. Without the Page Facility, the operator may view each page (one screen of the response) in sequence, using standard Intercomm facilities. If, after viewing the entire list, the terminal operator requires a name at the beginning of the list, then under normal conditions (without the Page Facility), that name is available only by reentering the same input message.

## 1.2 MESSAGE FLOW USING THE PAGE FACILITY

Figure 1 illustrates message flow using the Page Facility in the Intercomm system. The numbers below correspond to the numbers in the illustration:

1. The operator enters an inquiry (internally identified, for example, by message 501).

2. The appropriate user subsystem is initiated; it then processes the message.

3. The subsystem passes each output message (in the response to message 501) to the Page service routine.

4. The Page Facility saves the entire response on the Page Data Set.

5. It enters the necessary control information in the Page Table (that is, response number, address of response on page queue, number of pages).

6. It then passes the first page of the output response to the Output Utility (for formatting) if necessary, then

7. The first page is passed to FESEND for queuing for the requesting terminal.

8. The first page is transmitted to the terminal operator.

NOTE: If the response message has a VMI of X'57' or X'67' (or is created via MMU), the first page is passed directly to FESEND instead of via the Output Utility.

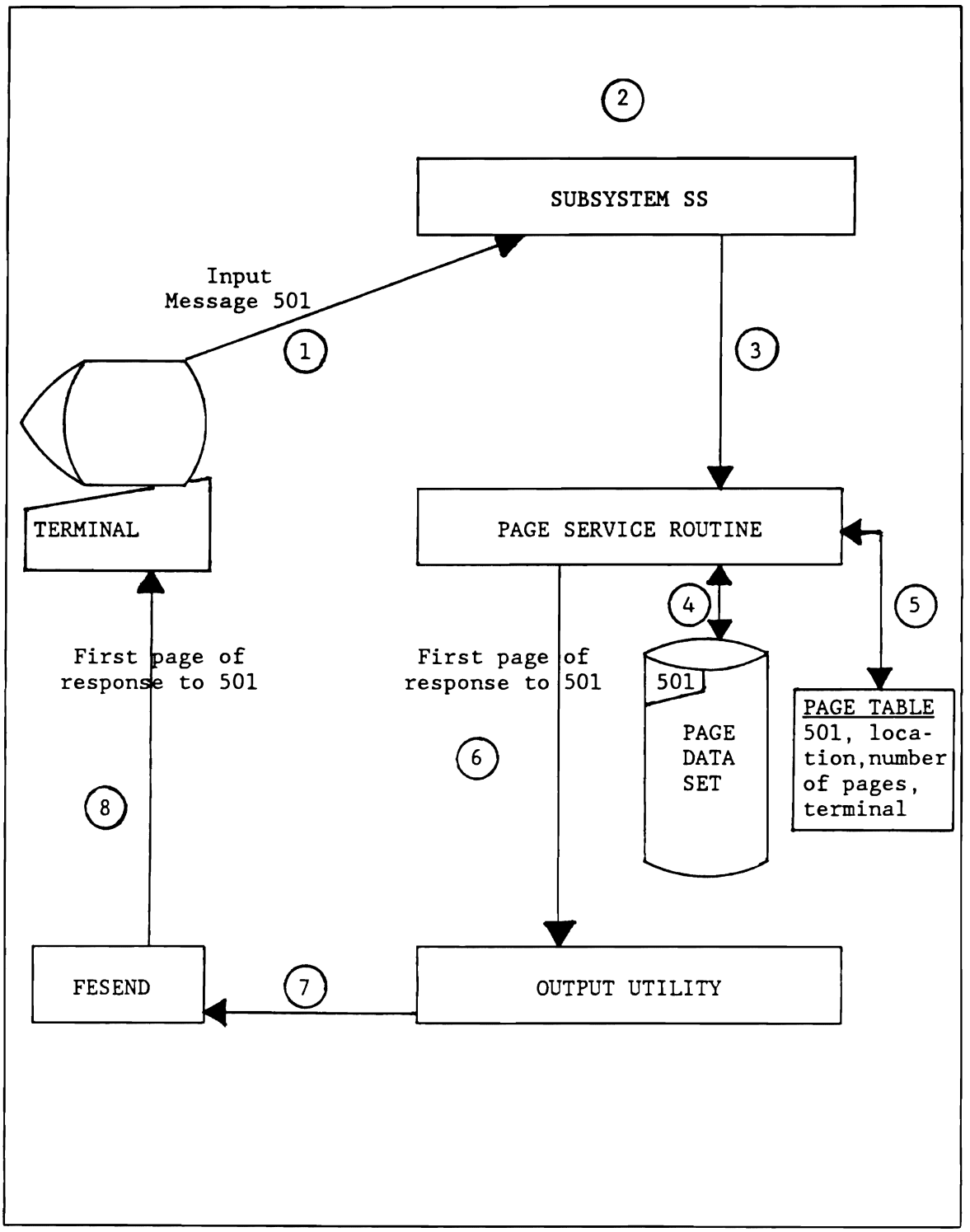


Figure 1. Message Flow Using Intercomm Page Facility

## 1.3 THE PAGE COMMAND

The terminal operator examines the first page of the response to message 501; then by entering different PAGE commands, he can effectively browse through the group of messages in the response saved by the Page service routine on the Page Data Set.

Figure 2 illustrates the message flow when the terminal operator issues a PAGE command. The numbers below refer to the numbers in Figure 2.

1. The terminal operator enters a PAGE command (for example, to request the next page of the response), which is passed directly to the Intercomm Page subsystem (PAGEMSG).

2. PAGEMSG retrieves the requested page from the response to message 501 on the Page Data Set (using the terminal-id to find the Page Table entry).

3. PAGEMSG passes the message to the Output Utility.

4. Output passes the message to FESEND for queuing for the requesting terminal.

5. The message is transmitted to the terminal.

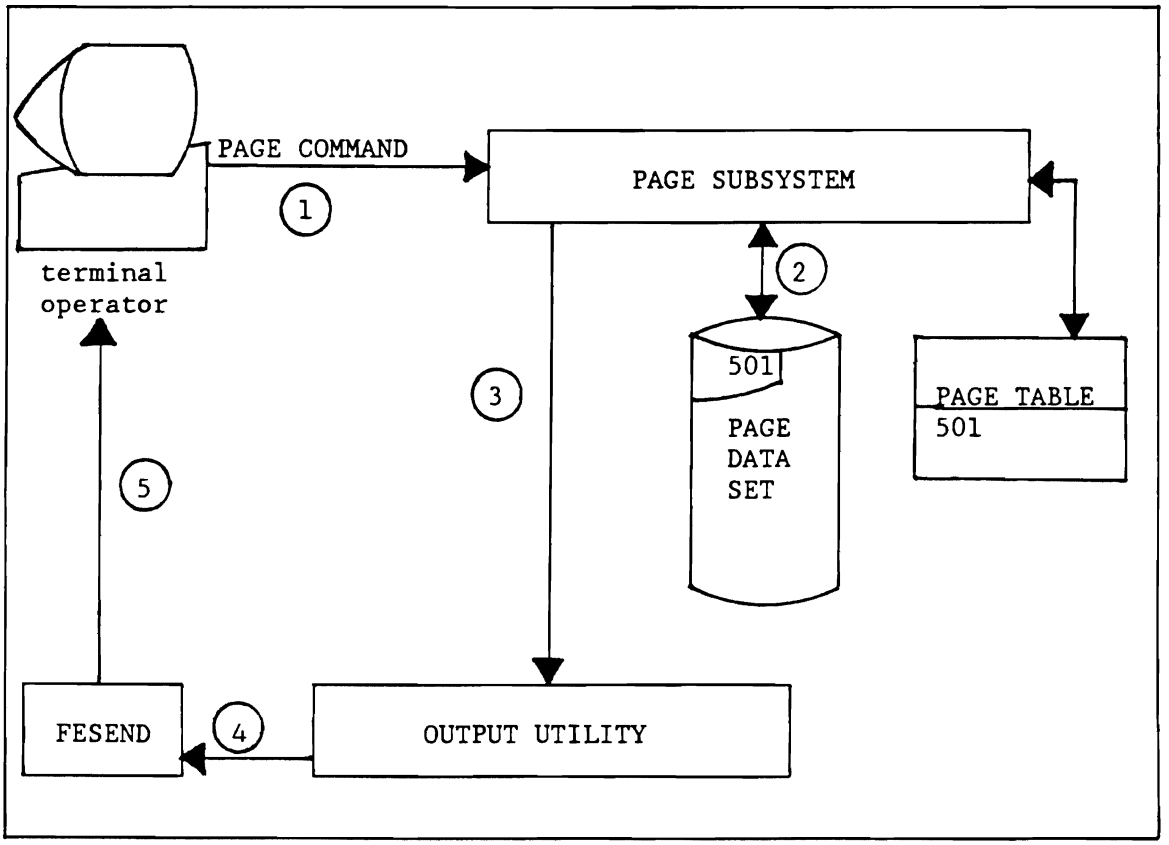


Figure 2. Message Flow When Issuing a PAGE Command

NOTE: Fully formatted (VMI=X'67') and preformatted (VMI=X'57') messages are passed directly to FESEND instead of to the Output Utility.

If the operator has completed viewing the response to 501, a new inquiry message can be entered. When the Page service routine recognizes a new input message number for that terminal, it assumes the previous response for that terminal is no longer required and deletes it from the Page Data Set. Otherwise, if the operator requires the response to 501 for future reference from that terminal, the SAVE command will save the pointers to that response in the Page Table; a subsequent response request will be concatenated to it. Thus, several responses for the same terminal may be saved for subsequent browsing (if there is room on the Page Data Set).

## 1.4 SUMMARY

There are six components involved in the Page Facility:

- The user application subsystem, which is designed in advance to include logic for developing a multiscreen output response to one input message from a visual display terminal.
- The Page service routine (PAGE), which is called by a user application subsystem or by MMU (MAPEND) for each output message of a multiscreen response.
- Terminal operator commands, which are entered to retrieve a page of a response or to save a set of pages, or to delete a response previously saved.
- The Page subsystem (PAGEMSG), which processes terminal operator commands.
- The Page Data Set. This is a BDAM file associated with each terminal for storing responses for operators to retrieve via commands. (The first message of a response is also automatically sent to the terminal). Several terminals may share the same physical data set; the term Page Data Set refers to a logical data set per terminal. More than one physical data set may be used as the number of terminals using Page grows.
- The Page Table (PAGETBL). This is a core-resident table used by both PAGE and PAGEMSG, with one entry for each terminal which may enter messages resulting in multiscreen (paged) output. This table defines the area of the specific BDAM data set to be used for the terminal's responses on the file, and has space for internal control information.

# Chapter 2

# APPLICATION PROGRAM INTERFACE

## 2.1 MESSAGE HEADER

As with all application subsystems operating under Intercomm, subsystems using the Page Facility must construct an output message header. Each input message, when it arrives from a terminal, is prefixed with a standard Intercomm message header which is constructed by the Front End. Output messages created directly by the subsystem, or indirectly via MMU, must be prefixed with this same message header. For the subsystem, this is usually accomplished by copying the input message header to the output message header area. However, certain fields must be adjusted to reflect the new status of the message. If the subsystem requests MMU to pass the formatted output messages directly to the Page Facility, the following discussion of the message header does not apply.

Figure 3 details the names, formats and contents of all the fields in the message header. The message header is constructed as if it were being passed to the Output Utility via MSGCOL or COBPUT: for example, MSGHRSCH=X'00' and MSGHRSC=C'U' indicating Output Utility formatting required and MSGHVMI=X'50' indicating a variable text format. The application programmer should refer to the appropriate Intercomm Programmers Guide for further information on the content of these fields and other message header and text processing options. If the subsystem preformats the message (Output Utility processing not needed), set MSGHVMI=X'57'.

Two other fields of particular importance to the Page Facility are the MSGHMMN and MSGHLEN fields. Every message passed to PAGE is required to contain in its message header the same MSGHMMN as the input transaction. This field is the Intercomm internal message number and the Page service routine compares this message number to the previous message number it processed for the terminal named in the MSGHTID field. When PAGE receives a different message number for a given terminal, it assumes that the previous group of messages it saved on the Page Data Set for that terminal is no longer required and may be overwritten.

A terminal operator can override the message number test by entering the SAVE command for the previous response. An application subsystem can bypass the test by inserting X'FFFFFF' in the MSGHMMN field. However, this latter procedure may cause interleaving of the message responses for that terminal on the Page Data Set rather than creating unique transaction responses in a multithread environment.

The other field in the message header of importance to the Page Facility is the MSGHLEN field. If a paged message is sent to the Output Utility in item code-length-data format, that is, variable text (VMI=50) format, the MSGHLEN field must be the exact length of the

message. This is required because the Page Facility appends an item code 252 to all item code-length-data format messages. Item code 252 denotes a three-character field containing the page number assigned to the message by Page. Thus, the Output Format Table coded for this message can control the location on the screen layout for the display of a page number. The page number may be ignored if not appropriate for a particular screen or application (do not code an ITEM macro for code 252 in the OFT).

| Field Name | Length | Description |
|---|---|---|
| MSGHLEN | 2 | Exact length of message including header (binary number) |
| MSGHQPR | 1 | Segment identification |
| MSGHRSCH | 1 | High-order receiving subsystem code |
| MSGHRSC | 1 | Low-order receiving subsystem code |
| MSGHSSC | 1 | Low-order sending subsystem code |
| MSGHMMN | 3 | Monitor message number assigned by Message Collection on input message (binary number) |
| MSGHDAT | 6 | Julian date (YY.DDD) assigned by LOGPUT |
| MSGHTIM | 8 | Time stamp (HHMMSSTH) assigned by LOGPUT |
| MSGHTID | 5 | Terminal identification (originating terminal) |
| MSGHCON | 2 | Reserved area |
| MSGHFLGS | 2 | Message indicator flags |
| MSGHBMN | 3 | Front End message number-Rel. 10 (binary) |
| MSGHSSCH | 1 | High-order sending subsystem code |
| MSGHUSR | 1 | (available to user) |
| MSGHBMN | 2 | Front End message number-Rel. 9 (binary) |
| MSGHLOG | 1 | Log code |
| MSGHBLK | 1 | Reserved |
| MSGHVMI | 1 | Verb/message identifier interpreted by receiving subsystem (Output), as required, and by FESEND/PAGE |

Figure 3. Message Header Fields

## 2.2 SUBSYSTEM INTERFACE VIA THE OUTPUT UTILITY

The following applies only to subsystems which require the Output Utility for formatting output messages. If the message is preformatted (VMI=X'57' or X'67'), the Page Facility passes the message directly to FESEND. For subsystems using MMU, see Section 2.3.

An application subsystem invokes the Page Facility whenever it has an output message that needs to be saved or paged. Instead of calling MSGCOL or COBPUT to queue the message for the Output Utility, the application calls the Page Facility, which saves the message on the Page Data Set and subsequently sends it to the Output Utility (or FESEND, depending on the VMI code) when requested by the terminal operator. The Page Facility is called for each page of the message generated by the subsystem.

On return from the Page Facility, the subsystem must test a return code from the Page service routine to determine the result of each request to add a message to the Page Data Set. Two conditions need be tested:

1. The result of queuing (applies only to the first message of a response); that is, was the message successfully passed to the Output Utility via MSGCOL or COBPUT, or successfully queued via FESEND?

2. The result of adding the first and subsequent messages to the Page Data Set.

The Page return code is a fullword binary value, except in the case of unsuccessful queuing of the first message of a response. In this case the leftmost two bytes of the word contain the MSGCOL (FESEND) or COBPUT return code. Typical subsystem logic is diagrammed in Figure 4. MSGCOL (FESEND) and COBPUT return codes and suggested recovery action are listed in Figure 5. Page return codes and suggested recovery action are listed in Figure 6. Note that in the case of unsuccessful queuing of the first message of a response, that message is not added to the Page Data Set.

An application subsystem can send PAGE and SAVE commands to the Page subsystem to recover from certain error conditions instead of leaving the responsibility to the terminal operator. For example, in the event of PAGE DATA SET FULL error conditions, the subsystem calling Page might send a message to the Page subsystem to terminate the current response, assuming the condition occurred in the middle of logic creating several pages. An error message should also be sent to the terminal operator as well, indicating that the response was terminated. The first page may have already been received at the terminal, however, because of elapsed time between creation of the first page and the page which resulted in the data set full condition. Messages are routed to the Page subsystem by filling in the message header receiving subsystem code as defined in the Subsystem Control Table entry for PAGEMSG (the Intercomm-supplied definition specifies MSGHRSCH=X'00', MSGHRSC=C'P' but may be varied by an installation).

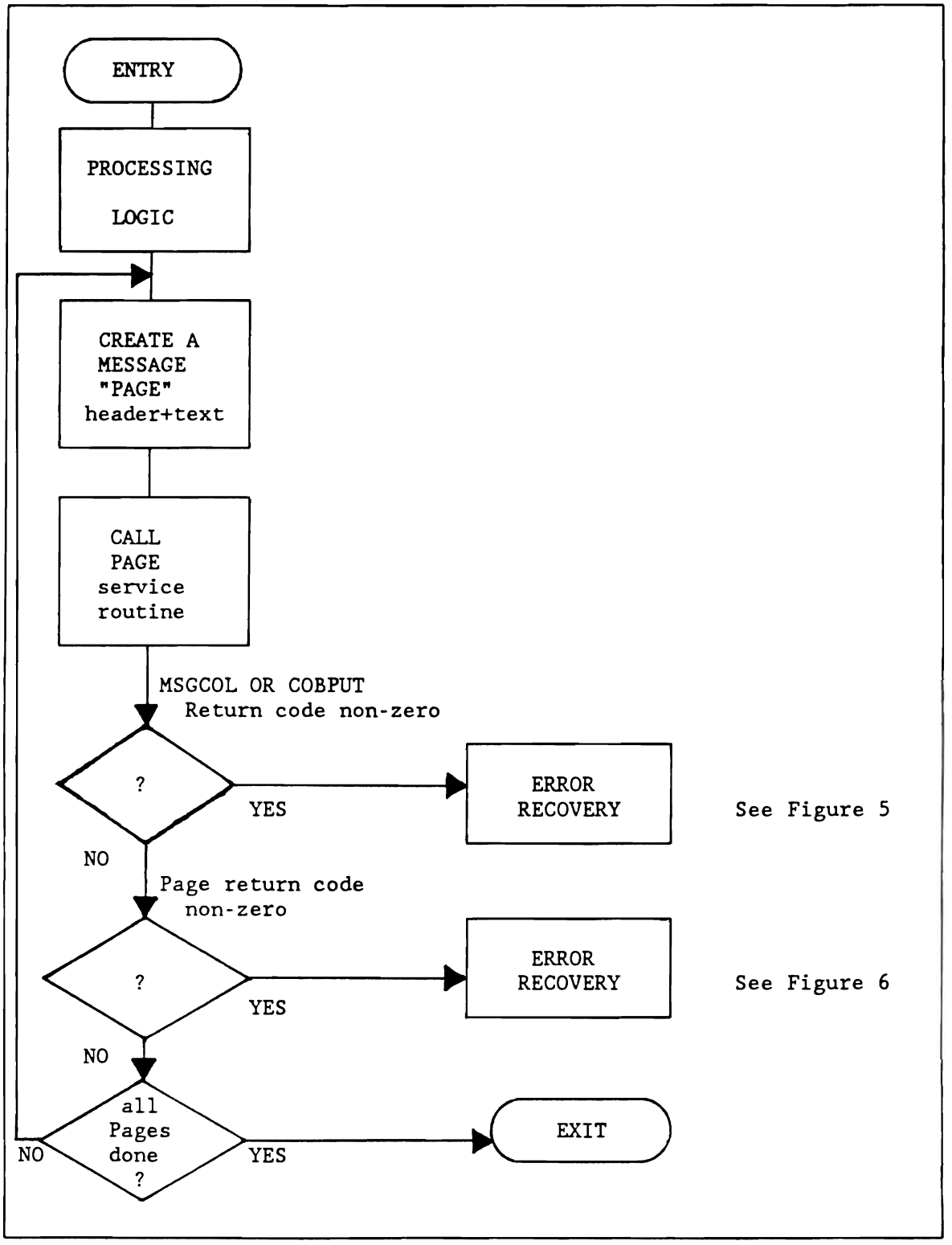


Figure 4. Typical Subsystem Logic Using PAGE

| MSGCOL/FESEND Return Code (binary) | COBPUT Return Code (character) | Meaning |
|---|---|---|
| 0 | 00 | Message queued successfully. |
| N/A | 02 | Item code, length, or line number greater than 255 in variable character data item prefix. |
| 4 | 04 | No room on terminal/Output Utility queue - an entry was made on the system log. |
| 8 | 08 | No core for disk queue I/O area, or to copy message. |
| N/A | 10 | N or R omitted in variable character data item prefix. |
| 12 | 12 | I/O error on disk queue. |
| 16 | 16 | Invalid subsystem code (MSGCOL/COBPUT for message switching) or invalid terminal-id (FESEND) - an entry was made on the system log. |
| 24 | N/A | FESEND called because VMI=X'57' or X'67': invalid message header - review logic to copy input message header. |
| N/A | 28 | DVASN could not assign a device (on first segment of multisegmented messages only). |

Figure 5. MSGCOL/FESEND and COBPUT Return Codes

| PAGE Return Code binary/hex | Meaning | Recovery Action |
|---|---|---|
| 0/00 | Message added to Page Data Set. If first message of response, it was successfully queued for the terminal. | None |
| 4/04 | Message is longer than the Page Data Set block size.* | Page data set must be reformatted or application logic changed if message length incorrect. |
| 8/08 | Insufficient core for Page to complete processing.* | Effect a delay (I/O or time) and retry. |
| 12/0C | I/O error on Page Data Set.* | Unrecoverable hardware error or 'Select' rejected. |
| 16/10 | Page Data Set is full.* | Page data set must be enlarged or operator should enter command to delete previous response(s). |
| 18/12 | No Page request queue elements (internal control blocks) available.* | See 8 above. Increase number of queue elements (cf. Page Table). |
| 20/14 | No Page Data Set defined for terminal named in message header (MSGHTID).* | Page Table (PAGETBL) must be revised. |
| 24/18 | RBN out of range.* | Page data set allocation too small for number of terminals using that data set. |
| *Message not added to Page Data Set | | |

Figure 6. PAGE Return Codes

NOTE: a return code value of 36 (binary) or X'24' or higher is the FESEND/MSGCOL return code plus 32 (X'20') for Assembler Language callers (including MAPEND) only. Subtract 32 (X'20') from that value to determine MSGCOL/FESEND return code and see Figure 5 for meaning.

### 2.2.1 Assembler Language Coding

The Page service routine is be called using the CALL macro.

Coding format:

```
[symbol]  CALL   PAGE,(msgaddr),VL,MF=(E,list)
```

where msgaddr is the address of the message to be paged. The VL parameter is required on the CALL statement. The message must be in a separate storage area from that of the save/work area, and the storage area length must be that of the message (in MSGHLEN).

The possible return codes passed from PAGE to the Assembler Language coded caller in Register 15 are listed in Figure 6. The return code is a binary value in the low-order byte.

In addition to the return codes outlined in Figure 6, a return code greater than 32 (X'20') is also possible if MSGCOL or FESEND returned an error code to PAGE. Return codes from MSGCOL (or FESEND if VMI=X'67' or X'57') are listed in Figure 5. PAGE adds 32 to the return code to distinguish it from the Page return codes in Figure 6. A return code of X'24' (binary 36) means no room on queue. On return from PAGE, the message area no longer belongs to the caller, even if the return code is not zero. A new area must be acquired (via STORAGE macro) for each subsequent message to be formatted.

### 2.2.2 COBOL Coding

A reentrant OS/VS or VS II COBOL coded program calls the Page service routine via COBREENT, using the following coding format:

```
CALL 'COBREENT' USING PAGE, msg, page-return-code.
```

where msg and page-return-code must be in the caller's DWS (Dynamic Working Storage) area (see COBOL Programmers Guide):

PAGE
: is the label of a halfword 'computational' field containing the decimal value 92. This is an offset in the REENTSBS table for the address of the Page service routine (See ICOMSBS copy table in the COBOL Programmers Guide).

msg
: is the label of the message to be 'paged'.

page-return-code
: is a fullword in which PAGE will place a return code.

Nonreentrant COBOL subsystems can directly call the Page service routine, if compiled under OS/VS COBOL, using the following coding format:

```
CALL 'PAGE' USING msg, page-return-code.
```

Figure 6 outlines the possible binary (computational value) return codes from the Page Facility. In addition, COBPUT or FESEND may return an error code to the Page routine which will be stored in the two high-order bytes of the return code area. The possible return codes from COBPUT or FESEND are listed in Figure 5.

COBPUT return codes are numeric character values. To determine whether COBPUT returned an error code, the two high-order bytes of the page-return-code area should be checked for valid numeric characters, or a non-zero value from FESEND (if VMI was X'57' or X'67' and the Output Utility not used). Otherwise, the return code area should be checked for a fullword computational value from PAGE.

Figure 7 illustrates an example of the possible programming logic to check the return code area.

```
LINKAGE SECTION DEFINITION OF RETURN CODE:

      02   PGRTCD     PICTURE      S9(8)     COMPUTATIONAL.
      02   RETURNCD   REDEFINES     PGRTCD.
           04    COBPUTRC       PICTURE     XX.
           04    FILLER         PICTURE     XX.

PROCEDURE DIVISION LOGIC:

*CALL PAGE TO ADD MESSAGE TO PAGE DATA SET.

      CALL 'COBREENT' USING PAGE, MSG, PGRTCD.

*TEST COBPUT RETURN CODE FIRST -  CHARACTERS.

*COBPUT RETURN CODE ONLY RETURNED BY PAGE IN THE

*CASE OF QUEUING ERRORS FOR OUTPUT UTILITY.

      IF COBPUTRC IS NUMERIC GO TO COBPUT-ERR.

*TEST PAGE RETURN CODE NEXT - COMPUTATIONAL.

      IF PGRTCD = ZERO GO TO GOODRTN.
      .
      .      PAGE error processing
      .
 COBPUT-ERR.
      .
      .      COBPUT error processing
      .
 GOODRTN.
      .
      .
      .
```

Figure 7. Example of Error Checking Logic for COBOL Coding

### 2.2.3 PL/1 Coding

A PL/1 subsystem may call the Page service routine via the Intercomm supplied subroutine PMIPL1, using the following coding format:

```
CALL PMIPL1(page-routine-pointer,msg,page-return);
```

page-routine-pointer
: is a halfword fixed binary field containing the decimal value 92. This is an offset in the REENTSBS table for the address of the Page routine (See PENTRY codes in PL/1 Programmers Guide).

msg
: is the label of the message to be paged (declared character).

page-return
: is a fullword field (declared FIXED BIN(31)) in which PAGE will place a return code, as described for COBOL programs.

A PL/1-Optimizer program can call PAGE directly if a %INCLUDE statement is coded for PLIENTRY, or the Page service routine is declared with OPTIONS (ASM,INTER). Use the following coding format:

```
CALL PAGE(msg,page-return);
```

## 2.3 SUBSYSTEM INTERFACE VIA MMU

As described in Message Mapping Utilities, a subsystem may call MAPOUT to format each output message. Then, at MAPEND time, the subsystem may request via the MCW that all formatted output messages be passed to PAGE by MMU. If successful, the first message will be transmitted to the requesting terminal (PAGE calls FESEND). Page processing return codes for a MAPEND request are described in Message Mapping Utilities. PAGE and SAVE command processing is the same as for messages formatted via the Output Utility.

## 2.4 TESTING SUBSYSTEMS USING THE PAGE FACILITY

In order to test subsystems using the Page Facility, operation with terminals is almost always a necessity. Because of timing considerations, any Page commands entered as input messages in test mode may be processed prior to the messages input to the user subsystems which create the pages of a response, or before all created messages are stored on the Page Data Set.

The subsystem which creates pages of a response may certainly be tested in test mode to validate its logic for creating pages. It is only the Page commands which can not be used successfully in test mode.

Messages passed to the Page Facility are not logged, except for the first message of a response (logged when queued for the Output Utility, or the terminal via FESEND). Subsequent messages (pages) of a response are logged when retrieved form the terminal-associated Page Data Set via PAGE commands (log code 01 if passed to the Output Utility, log code F2 when queued for the requesting terminal). Note that messages written to a Page Data Set, but never retrieved, are therefore never logged. User entries can be made on the log (via the LOGPUT service routine) after creating each page (and before calling PAGE) for verification of the message format. (See the applicable Programmers Guides.)

# Chapter 3

# TERMINAL OPERATOR COMMANDS

## 3.1 PAGE REQUEST COMMANDS

When a terminal operator enters a transaction which will cause the Page Facility to be invoked, the Page routine automatically sends the first output message generated by the subsystem to FESEND or to the Output Utility (depending on the VMI code) which in turn passes the message to the Front End for transmission to the terminal.

The first output message is also saved on the Page Data Set, as are all additional output messages generated from that input transaction. To subsequently view any pages from this response, the terminal operator must enter the PAGE command in one of the following forms (see System Control Commands for syntax):

```
PAGE$[(N)[$(n)]]@
     [(P)  (1) ]
     (S$n      )
     (C        )
     (L        )
```

n
: represents a page number (Next/Previous/Specific)

N
: requests the next nth page (default: n=1)

P
: requests the previous nth page (default: n=1)

S
: requests specific page number n (n must be specified) calculated from the first page of the first response group for the requesting terminal.

C
: requests retransmission of the current page

L
: requests the last page (of current response group)

NOTE: When the first page of a response is sent to a screen, the other pages are still being written to the Page Data Set. Without a short wait, the operator may be unable to immediately access the last page of a response.

## 3.2 SAVE RESPONSE COMMAND

If and when the Page service routine receives a message to be paged which is part of a subsequent response for the same terminal, the Page Facility will delete the previous response unless a SAVE transaction has been received from the terminal operator. SAVE requests that the next response be chained to the previous response for that terminal. Multiple response groups can subsequently be accessed via PAGE commands as if created in one single response group. There are no parameters for the SAVE transaction:

```
SAVE@
```

## 3.3 PAGE REPORT COMMAND

The terminal operator can obtain a display describing the current utilization of the Page Data Set by his terminal via the command:

```
PAGE$REPORT@
```

The Page Report indicates the current use of a terminal's Page Data Set, allowing the operator to verify the number of pages for various responses previously saved and the number of pages in the current response (as accumulated). The resulting display is illustrated below:

```
P A G E  D A T A  S E T  U T I L I Z A T I O N

RESPONSE          NUMBER          NUMBER OF
 NUMBER         OF PAGES         FIRST PAGE

  001               3                1
  002              26                4
  003               7               30
   .                .                .
   .                .                .
   .                .                .
  nnn              nnn              nnn
```

## 3.4 RESPONSE TERMINATION COMMANDS

The terminal operator controls the utilization of his terminal's Page Data Set via the following termination transactions:

```
PAGE${TC}@
     {TH}
     {TA}
     {TL}
```

TC
requests termination of the response group being viewed.

TH
requests termination of all response groups except the first one.

TA
requests termination of all response groups (overrides previous SAVE requests).

TL
requests termination of only the last response group.

A report (see Section 3.3) is returned if the termination command is successfully processed.

NOTE: Do not use SAVE or report termination commands, or a new request command for other processing, until you are sure all pages of the current response have been written to the Page Data Set. Otherwise, interleaving/overwriting of pages may occur.

## 3.5 MESSAGES RETURNED TO THE TERMINAL

1. RESPONSES SAVED PER YOUR REQUEST
   The acknowledgement to the SAVE command.

2. THE REQUESTED PAGE COULD NOT BE SENT
   Issued when PAGEMSG subsystem could not successfully queue the retrieved page.

3. THIS TERMINAL NOT DEFINED AS A PAGING TERMINAL
   Displayed when the terminal using the Page Facility has not been defined in the Page Table.

4. PARAMETER INCORRECTLY SPECIFIED FOR PAGE VERB
   This message results from a syntax error on a PAGE command operand.

5. I/O ERROR OCCURRED WHILE RETRIEVING REQUESTED PAGE
   The READ operation on the Page Data Set was unsuccessful.

6. PAGE REQUESTED IS OUTSIDE THE ALLOWABLE RANGE
   The Page Subsystem attempted to READ a block from the Page Data Set which did not exist. This message normally results when a PAGE$N (next) command has been preceded by a display of the last page of the response(s). (This message can also occur when the number of RBNs in the Page data set created via CREATEGF (see Chapter 4) is less than the sum of the blocks required as defined in the Page Table, and the application has attempted to build too many pages of a response for the RBNs provided.)

7. THE REQUESTED PAGE IS NOT PART OF ANY RESPONSE
   The page number requested is invalid for the given response accumulated for the original message to a subsystem. Verify how many pages are actually contained in the response via the PAGE$REPORT@ command; and verify that DCB=(DSORG=DA,OPTCD=RF) is coded on the Page data set DD statement.

## 3.6 OPERATIONAL NOTES

### 3.6.1 Locking a Terminal to the PAGE Verb

If many Page commands are going to be entered, the operator may save keystrokes by entering:

LOCK$TPUxxxxx$PAGE@

This will lock the terminal xxxxx onto the PAGE verb and all subsequent messages entered from the terminal xxxxx will have PAGE$ prefixed as a verb. After entering the

LOCK$TPUxxxxx$PAGE@

command, the operator enters N to get the next page, or enters P$3 to get the page which is 3 pages prior to the current one, etc.

The operator continues in this fashion until he wishes to enter another verb. Then he enters:

UNLK$TPUxxxxx@

and is then free to enter any valid verb.

### 3.6.2 IBM 3270 Input Message Formats:

IBM 3270 terminal users have a possibility of three different Front End input formats with respect to Page commands:

- Standard positional input--unformatted screen--results in message text with:

  PAGE$N ($ is the system separator character)

- Standard positional input--formatted screen--displayed as a response to a Page command results in message text with:

  aPAGE$N (a is an SBA sequence--deleted by the Front End)

- Positional input--formatted screen--where the Page command modifiers are input into unprotected fields--results in message text with:

  aPAGEaN (a is an SBA sequence)

  which is supported only if the Edit Control Table specifies IBM 3270 input, that is, specific buffer addresses for the command modifiers (the SBA sequence preceding the verb is deleted by the Front End).

To preclude problems with entering PAGE commands in formatted 3270 screens, an installation may set up AID Key conversion sequences (as described in the SNA and BTAM Terminal Support Guides) to equate PF Keys (also code REPLACE=YES on the corresponding AIDDATA macro), or to equate PA Keys, to the most commonly used PAGE commands such as PAGE$N and PAGE$P.

# Chapter 4

# INSTALLATION

## 4.1 OVERVIEW

In addition to the user application subsystems and their associated table definitions, the following items are required to implement the Page Facility:

1. Table definitions for the Page commands and the Page subsystem

2. Table definitions for the Page data set(s), and execution of the off-line utility CREATEGF to format the Page data set(s)

3. Intercomm linkedit to include tables and programs for the Page Facility, and execution JCL to reference the Page data set(s)

Security and message restart considerations are also given.

## 4.2 DEFINING THE PAGE COMMANDS AND SUBSYSTEM

### 4.2.1 Front End Verb Table (BTVRBTB CSECT)

Entries are required in the Front End Verb Table for the PAGE and SAVE commands and to specify the associated command processing subsystem code. The following entries are supplied in the released BTVRBTB on SYMREL:

```
PAGE     BTVERB     VERB=PAGE,SSC=P,EDIT=YES,CONV=36000
SAVE     BTVERB     VERB=SAVE,SSC=P,EDIT=YES,CONV=36000
```

Use of the CONV parameter marks the verb as conversational and prevents the operator from entering another input message until a response is received from that verb's processing subsystem.

No terminal may enter concurrent transactions which cause paging. This applies to user verbs and Page commands. The operator may be restricted to "conversational mode" by the CONV parameter definition (as shown above) for the PAGE, SAVE, and all affected user verbs in the Front End Verb Table (see the BTVERB macro in Basic System Macros for further details).

NOTE: If Intercomm is executing in Multiregion mode, the PAGE and SAVE commands may be edited before being queued for a satellite region if EDIT=BQ is added to the BTVERB macro and the Edit Control Table (PMIVERBS) is included in the control region linkedit (Edit Utility not used in Satellite Regions). See also Chapter 5.

### 4.2.2 Edit Control Table--PMIVERBS (VERBTBL CSECT)

Entries are supplied in the released Edit Control Table on SYMREL to define processing of the PAGE and SAVE verbs, as follows:

```
PAGE    VERB    PAGE,01,,2,FIX=YES,KEY=NO
        PARM    CDE,1,0,2,10000110
        PARM    NUM,2,2,2,00000100
SAVE    VERB    SAVE,E2,,1,FIX=YES,KEY=NO
        PARM    NUL,1,0,1,00000100
```

See also Section 3.6, "Operational Notes."

### 4.2.3 Subsystem Control Table--INTSCT (SCT CSECT)

An entry is required in the Subsystem Control Table (SCT) for the PAGE and SAVE commands processing subsystem (PAGEMSG) in the same region as the Page service routine. The subsystem may be resident or dynamically loaded. Entries are provided in the released (on SYMREL) INTSCT (single-region system) and SR1SCT (for a Satellite Region in Multiregion mode) as follows:

```
P       SYCTTBL  SUBC=P,                          subsystem code
                 LANG=RBAL,                       language
                 SBSP=PAGEMSG,                    entry point
                 TCTV=120,                        2-minute timeout
                 NUMCL=5,DFLN=PMIQUE,PCEN=5,      queues
                 MNCL=5,                          multi-threading
                 RESTART=NO                       no message restart
```

## 4.3 DEFINING THE PAGE DATA SET(S)

Each terminal which may input messages resulting in a paged response (from a user subsystem which uses the Page Facility) must have a Page Data Set defined. A Page Data Set is a logical entity per terminal; several terminals may share the same physical data set. There may be any combination of shared or dedicated Page data sets. Typically, an installation would use just one physical Page data set, shared by all terminals, unless there is a large difference in the message sizes for different terminals, and/or many terminals. For example, an installation with application-dedicated terminals would benefit with respect to disk space allocation if different physical Page data sets were defined, one for each type (screen or message size) of terminal.

### 4.3.1 The Page Table (PAGETBL CSECT)

The Page Table contains an entry for each terminal using the Page Facility, defining (via the PAGETBL macro):

- The physical Page data set by ddname
- The number of blocks (RBNS) to reserve for that terminal (the logical data set size)
- The physical data set block size

The Page Facility subroutine SRCHPTBL (Search Page Table) contains an illustrative Page Table as its first CSECT. The CSECT may be updated or a separate Page Table called PAGETBLE may be generated (with CSECT name PAGETBL) and assembled and then included in the linkedit prior to SRCHPTBL.

The distributed member SRCHPTBL contains:

```
PAGETBL     CSECT
            PAGETBL TPU=CNT01,DDNAME=PAGES,RBNS=50,BLKSIZE=1070
            PAGETBL TPU=PER01,DDNAME=PAGES,RBNS=50,BLKSIZE=1070
            PAGETBL TPU=PER02,DDNAME=PAGES,RBNS=50,BLKSIZE=1070
            PAGETBL RQES=20
```

The PAGETBL macro is described in detail below. The RQES parameter on the last PAGETBL macro defines the total number of internal control blocks available for use by the Page Facility. The user-coded Page Table must be delimited by an Assembler END statement, when assembled separately from SRCHPTBL.

### 4.3.2 The PAGETBL Macro

The PAGETBL macro is used to create the Page Table entries that furnish the Intercomm PAGEMSG subsystem and the Intercomm Page service routine with the information required by these routines.

The form of the PAGETBL macro is as follows:

```
(blank)        PAGETBL     Terminal defining parameters:

                           RBNS={number-of-pages-to-reserve}
                                {1                         }

                           ,TPU=terminal-id

                           Page Data Set parameters:

                           ,BLKSIZE={Page-data-set-blocksize}
                                    {1080                   }

                           ,DDNAME=Page-data-set-ddname

                           Control block defining parameter:

                           [RQES=#-Response-Queue-Elements]
```

BLKSIZE
: specifies the block size of the Page data set defined by the DDNAME parameter. This block size must be large enough to hold the largest message (including message header) to be sent to the terminal being defined. The default is 1080. This parameter must be the same for all TPUs pointing to the same data set. The maximum block size is 32760, however note that messages are neither blocked, nor spanned across blocks.

DDNAME
: specifies the ddname of the Page data set. Each terminal may have its own data set or may share a data set with other terminals. All PAGETBL macros for terminals sharing a data set must be coded consecutively (be grouped together) under Release 9 (not required under Release 10). The maximum total RBNs for each shared data set is 32,766. Under Release 10 a maximum of 50 different Page data sets may be used; there is no limit under Release 9.

RBNS
: specifies the maximum number of pages (across all response groups) which may be saved for this terminal. This number may be all, or a portion, of the Page data set defined by the DDNAME parameter. The default is 1. The maximum is 32766.

RQES

specifies the number of Response Queue Elements to be generated. This number will be the total number of responses saved for all paging terminals at any given time. As an absolute minimum, this number must exceed (by at least one) the number of terminals defined as paging terminals. The maximum (up to 32767) would be the maximum possible saved responses for any one terminal times the number of PAGETBL macros in the Page Table, which would be necessary only if all paging terminals will be using the Page Facility concurrently. This parameter is only coded for the last PAGETBL macro, and also signifies the end of the table.

TPU

specifies the five-character Intercomm name of the terminal which is being defined as a paging terminal.

### 4.3.3 Page Table Data Set Allocation Report (Release 10 only)

When the Page Table (Csect PAGETBL) is assembled under Release 10, a report is generated at the end on the number of RBNs requested per defined Page data set ddname along with the defined BLKSIZE of the data set. This report should be used for creating new data sets and/or changing existing data sets as the Page Table is changed. If a PRINT NOGEN statement exists among the PAGETBL macros, ensure a PRINT GEN statement is placed before the final macro (with RQES parameter) so that the report may be printed. A sample report follows:

```
***          ACCUMULATED RBNS PER PAGE DATA SET          ***

***     DEFINED DDNAME          RBNS       BLKSIZE      ***

            PAGES                150          4096

            PAGE1                360          2048

            PAGE2                280          3072
```

### 4.3.4 Preformatting the Page Data Set(s)

The Intercomm utility CREATEGF can be used to preformat a Page data set as a BDAM file. See the Operating Reference Manual for details. The following sample JCL might be used to create a data set with ddname PAGES:

```
//          EXEC      PGM=CREATEGF
//STEPLIB   DD        DSN=INT.MODREL,DISP=SHR
//SYSPRINT  DD        SYSOUT=A
//SYSSNAP   DD        SYSOUT=A
//SYSUDUMP  DD        SYSOUT=A
//PAGES     DD        DSN=INT.PAGES,DISP=(,CATLG,DELETE),
//                    SPACE=(bbbb,(150)),
//                    DCB=(DSORG=DA,BLKSIZE=bbbb),
//                    UNIT=3330,VOL=SER=123456
//SYSIN     DD *
F PAGES     0150
```

Where bbbb is the block size defined for this Page Facility data set via PAGETBL macros. The SYSIN control card for PAGES specifies the number of blocks to create and must be greater than or equal to the sum of the RBNS operands of the PAGETBL macros specifying DDNAME=PAGES.

NOTE: When writing a page to a Page Data Set, the Page Facility issues an enqueue on the associated terminal-ID. If an enqueue time-out (Snap 114) occurs, it usually indicates contention accessing the associated Page data set, or disk queuing problems for the Output Utility or for associated terminal when queuing the first message of a response. The enqueue timeout value is taken from that coded for the NQTIM parameter on the SPALIST macro for the region (see Basic System Macros). It may be necessary to increase that time value.

## 4.4 DEFINING THE PAGE ROUTINE USER EXIT--USRPAGEX

The PAGE service routine, having determined that there is a Page Table entry for the requesting terminal, and that the message (page) is not too big to be placed on the associated Page Data Set, will call a User Exit (USRPAGEX) to validate or reject the message. The Exit may determine further whether the message should be added to the current or a new response for the terminal in the message header, or is a continuation of an earlier response for that terminal.

The parameter list (address in Register 1) to this routine contains the address of the message (page) to be added, followed by the address of the Page Table entry for the terminal defined in the message header.

Should the message be accepted for adding to the current response, or for creating a new response, then the routine must return a return code of 0 (zero) in Register 15. Any other return code requests rejecting the message (page), and should be a validly defined PAGE return code.

This exit routine must be single threaded, and may not give up control to the Intercomm Dispatcher; standard linkage processing must be used.

A sample of this exit routine is provided (USRPAGEX on SYMREL) and performs the following two checks:

1) Will verify that the MMN of the message (page) is equal to, or higher than, that of the last message (page) added for the terminal. If this is not the case, then the message will be rejected with the PAGE return code 16 (X'10') in Register 15. This will ensure that should multithreading occur for the same terminal in subsystems using PAGE, then only the latest set of pages will be processed, unlike now where interleaving occurs.

2) Should ESS or Basic Security (with signon) be implemented for the terminal in the message header, then it will ensure that a user is currently signed-on. If not, then the message will be rejected with the PAGE return code of 16 (X'10'), in case the user signs off before all the pages of the current response are added.

A user written version of this exit must COPY PGEDSECT and declare a USING PAGEMASK,Rn to have addressability to the passed Page Table entry for the terminal in the message header, and declare a USING statement for the label of a message header DSECT statement which is followed by the MSGHDR macro (see Basic System Macros) to address the terminal-id field (MSGHTID) in the passed message header. Because the exit may not give up control to the Dispatcher, a local save area may be used (see the sample USRPAGEX), use of the LINKAGE or SUBLINK and RTNLINK macros to acquire and free a save area is not necessary.

## 4.5 INTERCOMM LINKEDIT and EXECUTION JCL

See the Installation Guide (ICOMGEN macro) and Basic System Macros (ICOMLINK macro) for automated Page Facility installation. Then, ensure that the Intercomm Linkedit contains INCLUDE statements for the following Page Facility, and related, routines and tables:

- Updated BTVRBTB (Front End Verb Table) (Control Region only in Multiregion mode)
- Updated PMIVERBS (VERBTBL--Edit Control Table) (in same region as Edit Utility routines in Multiregion mode)
- Updated SCT (Subsystem Control Table--entry for PAGEMSG)
- New PAGETBLE (unless SRCHPTBL Page Table updated)
- PAGE Facility Routines:
  INCLUDE SYSLIB(SRCHPTBL)
  INCLUDE SYSLIB(PAGE)
  INCLUDE SYSLIB(USRPAGEX)--if used/coded
- Page Command Subsystem
  INCLUDE SYSLIB(PAGEMSG)--resident; omit from linkedit if dynamically loaded (and change SCT entry to specify LOADNAME=PAGEMSG, not SBSP=PAGEMSG)
- OFTs used for error messages by PAGEMSG Subsystem (in same region as Output Utility) and for subsystem generated messages (if any):
  RPT00045 and all OFTs used by the Edit and Output Utilities (see Messages and Codes) must be INCLUDEd after PMIRCNTB, or must be available via the RCT000 data set
- Intercomm Edit and Output Utility routines (may be only in Control Region in Multiregion mode--see PMIVERBS above).

In each region using the Page Facility, the following DD statement should be added to the Intercomm execution JCL for each Page data set (change ddname and DSN parameter as applicable):

```
//PAGES    DD  DSN=INT.PAGES,DISP=SHR,DCB=(DSORG=DA,OPTCD=RF)
```

Also, a FAR statement (see Operating Reference Manual) should be defined for each Page data set (change PAGES ddname as applicable), for example:

```
PAGES,OPEN=BASIC,ICOMBDAMXCTRL
```

## 4.6 INTERCOMM RESTART AND THE PAGE FACILITY

Message responses saved under one Intercomm execution are not preserved for the next Intercomm execution. Each response must be recreated after an Intercomm restart or new startup. Do not specify Page data sets for file recovery, nor page creation subsystems for message restart.

## 4.7 SECURITY CONTROL AND THE PAGE FACILITY

When a terminal operator has SAVE'd responses on the terminal's Page Data Set, it is usually undesirable to allow the next operator to use that terminal to see the responses created by the previous operator. If a security system executes under Intercomm, then a user exit at sign-off time should be coded to always send a PAGE$TA command message via MSGCOL to the PAGEMSG subsystem to delete any leftover responses. The header of this message must have a non-zero value in MSGHSSC to indicate an internal message (no response desired). If a security system is not used, but VTAM is, then a VTAM HALT user exit should be coded to send the PAGE$TA command at logoff time (see SNA Terminal Support Guide).

The Release 10 supplied user exit SECUEXIT for ESS (see Extended Security System) has comments for inserting such code after the label CHKPAGE and is called for a sign-off and/or session timeout. Use MSGHSSC=C'E' in the header. In association with SECUEXIT, the supplied LUCUR ESS exit routine for VTAM LU's at HALT time (during logoff via internal or external SPLU command) will ensure an ESS sign-off, if needed. If file security is implemented under ESS, the ddnames for the Page data sets should start with the letters INT (or PMI) to be exempt from file security processing. However, if the ddnames start with the letters PAGE (or some other common prefix), insert an entry for the prefix in the exempt file list in INTSEC00 after the statement with the label XMPTFILE as follows:

```
        DC      AL1(n),CL8'name-prefix'
```

where n is the true length of the name-prefix minus 1 (that is, n=3 for the prefix PAGE), and name-prefix is the assigned first letters of all Page data set names (that is, PAGE). Therefore for the prefix PAGE, the inserted statement would be: DC AL1(3),CL8'PAGE'. The numeric length values (following the L) are required and may not be changed.

For Basic Security, see the description of the USRSGNOF user-coded exit for sign-off cleanup processing in the Operating Reference Manual. This exit may be coded to internally queue a PAGE$TA message as described above.

For both ESS and Basic Security, the Page Facility supplied USRPAGEX user exit routine ensures messages still being generated for an earlier response are not interleaved with those for a new response, and that a user is still signed on at the terminal before allowing adding a page to the current response or creation of a new response.

# Chapter 5

## USING THE PAGE FACILITY WITH THE MULTIREGION SUPPORT FACILITY

To use the Page Facility in a Multiregion environment, three choices are available as follows:

- Optimize data set storage by confining all Page Facility access and command processing to one region (control or satellite)
- Without RAP implemented, use different verbs (and different PMIVERBS entries) and unique associated subsystem codes for PAGE and SAVE command processing in each region (add to region's SCT and to the RDT).
- With RAP implemented, Page processing may be defined in every region and use the same verbs and one PMIVERBS set of entries. The RDT must have the same entry for the Page subsystem coded in each region.

Each region where Page processing is needed must have unique data sets assigned. Page data sets may not be shared across satellite regions, nor with the control region. It is also necessary that the PAGE and SAVE commands for the current terminal processing must access the data set to which the messages for that terminal were originally written. That is, the input command messages must be passed to the region that processed the original user message which resulted in subsystem access to the Page Facility for that terminal (terminal must remain locked to the same region).

The PAGETBLE, PAGE, PAGEMSG, SRCHPTBL and USRPAGEX (if used/coded) modules must be accessible in each region which will use the Page Facility. The same Page Table may be used in each region, or a limited version may be used in a test region. While PAGETBLE may be the same for all regions, the referenced data sets must be unique to each region. Editing may be performed in the control region only (before queuing) or each region may have the Edit Utility and PMIVERBS entries for the Page Facility commands. Without RAP, but with unique verbs used for each region, the Edit Control Table entries in PMIVERBS must use the exact coding described in Section 4.2.2; only the verbs may be changed. The Output Utility and the reports used by PAGEMSG, Edit and Output, should be only in the control region. The PAGEMSG subsystem must have a SYCTTBL entry in the Subsystem Control Table and a SUBSYS entry in the Region Descriptor Table (RDT) for each applicable region. See Multiregion Support Facility for further considerations.

# INDEX

# INDEX