Transaction Processing Facility

# Data Communications Services Reference

*Version 4 Release 1*

SH31-0145-03

Transaction Processing Facility

# Data Communications Services Reference

*Version 4 Release 1*

SH31-0145-03

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

**Fourth Edition (June 2002)**

This is a major revision of, and obsoletes, SH31-0145-02 and all associated technical newsletters.

This edition applies to Version 4 Release 1 Modification Level 0 of IBM Transaction Processing Facility, program number 5748-T14, and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

IBM welcomes your comments. Address your comments to:

IBM Corporation
TPF Systems Information Development
Mail Station P923
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

# Tables

# Notices

References in this book 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 in this book is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

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:

IBM Corporation
Department 830A
Mail Drop P131
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Any pointers in this book to non-IBM Web sites are provided for convenience only and do not in any way serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this book or accessed through an IBM Web site that is mentioned in this book.

## Trademarks

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

- APPN
- Advanced Peer-to-Peer Networking
- CICS
- EOCF/2
- IBM
- PS/2
- System/370
- VTAM.

Other company, product, and service names may be trademarks or service marks of others.

# About This Book

This book describes the functions provided for the data communication area of the TPF system.

In this book, abbreviations are often used instead of spelled-out terms. Every term is spelled out at first mention followed by the all-caps abbreviation enclosed in parentheses; for example, Systems Network Architecture (SNA). Abbreviations are defined again at various intervals throughout the book. In addition, the majority of abbreviations and their definitions are listed in the master glossary in the *TPF Library Guide*.

## Who Should Read This Book

This book is intended for system programmers who are responsible for data communications support.

## Conventions Used in the TPF Library

The TPF library uses the following conventions:

| Conventions | Examples of Usage |
|---|---|
| *italic* | Used for important words and phrases. For example:<br>A *database* is a collection of data.<br>Used to represent variable information. For example:<br>Enter **ZFRST STATUS MODULE** *mod*, where *mod* is the module for which you want status. |
| **bold** | Used to represent text that you type. For example:<br>Enter **ZNALS HELP** to obtain help information for the ZNALS command.<br>Used to represent variable information in C language. For example:<br>**level** |
| `monospaced` | Used for messages and information that displays on a screen. For example:<br>`PROCESSING COMPLETED`<br>Used for C language functions. For example:<br>`maskc`<br>Used for examples. For example:<br>`maskc(MASKC_ENABLE, MASKC_IO);` |
| ***bold italic*** | Used for emphasis. For example:<br>You ***must*** type this command exactly as shown. |
| **Bold underscore** | Used to indicate the default in a list of options. For example:<br>**Keyword=OPTION1** \| **DEFAULT** |
| Vertical bar \| | Used to separate options in a list. (Also referred to as the OR symbol.) For example:<br>**Keyword=Option1** \| **Option2**<br>**Note:** Sometimes the vertical bar is used as a *pipe* (which allows you to pass the output of one process as input to another process). The library information will clearly explain whenever the vertical bar is used for this reason. |

| Conventions | Examples of Usage |
|---|---|
| CAPital LETters | Used to indicate valid abbreviations for keywords. For example:<br>KEYWord=*option* |
| Scale | Used to indicate the column location of input. The scale begins at column position 1. The plus sign (+) represents increments of 5 and the numerals represent increments of 10 on the scale. The first plus sign (+) represents column position 5; numeral 1 shows column position 10; numeral 2 shows column position 20 and so on. The following example shows the required text and column position for the image clear card.<br>`\|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...`<br>`LOADER  IMAGE CLEAR`<br>**Notes:**<br>1. The word LOADER must begin in column 1.<br>2. The word IMAGE must begin in column 10.<br>3. The word CLEAR must begin in column 16. |

## Related Information

A list of related information follows. For information on how to order or access any of this information, call your IBM representative.

## IBM Transaction Processing Facility (TPF) 4.1 Books

- *TPF ACF/SNA Data Communications Reference*, SH31-0168
- *TPF ACF/SNA Network Generation*, SH31-0131
- *TPF Concepts and Structures*, GH31-0139
- *TPF Database Reference*, SH31-0143
- *TPF General Macros*, SH31-0152
- *TPF Main Supervisor Reference*, SH31-0159
- *TPF Operations*, SH31-0162
- *TPF Program Development Support Reference*, SH31-0164
- *TPF System Generation*, SH31-0171
- *TPF System Installation Support Reference*, SH31-0149.

## Miscellaneous IBM Books

- *EP for 3705 Generation and Utilities Guide*, SC30-3242
- *General Information - Binary Synchronous Communications*, GA27-3004
- *IBM Extended Operations Console Facility/2 System Administrator's Guide*, SH31-0105
- *NetView Operation*, SC30-3364
- *NetView Command Lists*, SC30-3423
- *Target System Control Facility Customization and Programming*, GC28-1066
- *Target System Control Facility General Information*, GC28-1063
- *Target System Control Facility Planning and Installation*, GC28-1065.

## Online Information

- *Messages (Online)*
- *Messages (System Error and Offline)*.

## How to Send Your Comments

Your feedback is important in helping to provide the most accurate and highest quality information. If you have any comments about this book or any other TPF information, use one of the methods that follow. Make sure you include the title and number of the book, the version of your product and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

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

- If you prefer to send your comments electronically, do either of the following:
  - Go to http://www.ibm.com/tpf/pubs/tpfpubs.htm.

    There you will find a link to a feedback page where you can enter and submit comments.
  - Send your comments by e-mail to tpfid@us.ibm.com
- If you prefer to send your comments by mail, address your comments to:

  IBM Corporation
  TPF Systems Information Development
  Mail Station P923
  2455 South Road
  Poughkeepsie, NY 12601-5400
  USA
- If you prefer to send your comments by FAX, use this number:
  - United States and Canada: 1 + 845 + 432 + 9788
  - Other countries: (international code) + 845 + 432 +9788

# Automated Operations Enabler

## Overview of Automated Operations

### What is Automated Operations?

Automated operations in TPF perform or assist in the execution of a large subset of tasks that operators traditionally perform. These tasks include monitoring and responding to messages as well as issuing commands.

### Why Should You Use Automated Operations?

Automated operations yield such benefits as:

- *Increasing system availability.* There is a reduction in the number of avoidable outages, and there is a reduction in the recovery and restart times from any outages. This leads to greater system availability.
- *Centralizing operations at one system.* One system can be set up as a focal point to monitor other system activity at other sites.
- *Removing constraints for system growth.* The complexity of operating various systems is reduced. Automation and standardization help alleviate these constraints.
- *Increasing reliability.* There are more uniform responses to error situations.

### How Does Automated Operations Benefit the Operator?

Automated operations relieve the operator of many mundane day-to-day tasks. TPF system can be automated in various ways, through its console interface.

The TPF system's implementation of automated operations allows for the filtering, suppression, and reformatting of select system messages. These select system messages can then be passed to an outboard automation platform where command processors and/or command lists can interpret the message and determine the appropriate command response based upon the message received.

Operators can respond faster to situations, while being able to focus on the real problems without having to worry about extraneous information, or having to issue trivial commands. In effect, the operator can now become a specialist in a streamlined data processing environment.

Additional operator benefits include:

- Simplifying operator tasks
- Increasing operator productivity
- Improving message traffic, and
- Decreasing human error.

#### Message Filtering/Suppression

Message filtering allows you to suppress trivial or informational messages so that the operator only sees messages that require operator intervention and action.

#### Message Reformatting

Messages may be reformatted for:

- Easier reading
- Easier interpretation
- Greater identification about where the messages are coming from, and

- Routing of the messages to the appropriate terminal.

## Terms and Concepts You Need to Know

You should read this section to become familiar with the following terms and concepts.

| Term/Concept | Definition |
|---|---|
| **automated operation** | In a generic sense, an automated operation is a computer program designed to assist a computer operator in the management of daily tasks. Automated operations help in the human interaction with any event coming from any system or network component. |
| **automation procedure** | A sequence of commands that can be a command list or a command processor written in a high-level language such as REXX. |
| **command list** | This is a sequential list of commands and control statements designed to perform a specific function. When the name assigned to this list is invoked as a command, the commands in the list are executed. |
| **command processor** | This is a high-level language program that is written to run under NetView. The function of this program is to perform an operation specified by a command (similar to a command list), but with much higher speed and flexibility. |
| **external automation** | When the target system is down, external automation is used to allow another computer system to perform an initial program load (IPL) and restart the target system. The TPF system is the target system being controlled. |
| **focal point system** | The controlling system where Target System Control Facility (TSCF) (5688-139) runs. (TSCF is described on page 3.) The focal point system is an MVS system that includes the NetView program. Contrast with *target system.* |
| **internal automation** | A situation where automation consists of programs inside the target system. Internal automation is only valid if the target system is functioning. When the target system is down, internal automation is also down. |
| **non-steady state** | The point in time when the operating system is being restarted when normal communications facilities are not available (only primitive console operations are available). For the TPF system, this is from IPL until the system reaches NORM or computer room agent set (CRAS) state. |
| **SNA character string** | The System Network Architecture (SNA) character string (SCS) is a data stream used by logical unit (LU) type 1 sessions. This data stream consists of EBCDIC control characters optionally mixed with end user data. |

| | |
|---|---|
| **steady state** | Normal system operations, along with full communication facilities running. This is when message volumes may require a high-performance communications link to the focal point. |
| **target system** | This is the computer system being monitored and controlled. In our scheme of things, the target system is the TPF system. |
| **Terminal Access Facility** | The Terminal Access Facility (TAF) is an option of NetView that allows a NetView operator to emulate a CICS, TSO/E, or IMS SNA LU type 1 or type 2 terminal. |

# TPF Automated Operations Enabler Requirements

## Implementing Automated Operations in the TPF Environment

The TPF system's implementation of automated operations supports using Target System Control Facility (TSCF) and/or NetView's Terminal Access Facility (TAF) in order to accomplish this.

See "Primary Areas of the TPF System Affected by Automated Operations" on page 6 for more detailed information about areas affected within the TPF system.

### Target System Control Facility (TSCF)

The Target System Control Facility (TSCF) gives you the ability to configure, initialize, shut down and monitor target systems from a focal point system. Customer written automation procedures assist the human (NetView operator) in controlling the target system.

TSCF allows you to:

- Perform a power-on reset of target system processors
- Perform an IPL of a target system
- Set the target system's time-of-day (TOD) clock
- Synchronize the clocks on all target systems
- Configure the target system (hardware only)
- Detect and respond to target system messages
- Shut down the target system.

TSCF consists of host and PS/2 code, and therefore requires a PS/2 computer as part of its configuration. The PS/2 computer can be connected to the focal point systems using an IBM token-ring local area network (LAN) or by a synchronous data link control (SDLC) link.

Each target system requires two connections to a PS/2 computer; one is for the hardware system console and the other is for its operations system console (see Figure 1 on page 4). To hook up the hardware system console, see *Target System Control Facility Planning and Installation*, GC28-1065. There are two ways to connect TSCF to the TPF system:

- The TPF system generated with native 3270 console support connects to TSCF through a 3270 connection card attached to a non-SNA terminal control unit
- The TPF system generated with 3215 console support needs an ASCII protocol converter and an ASCII connection card.

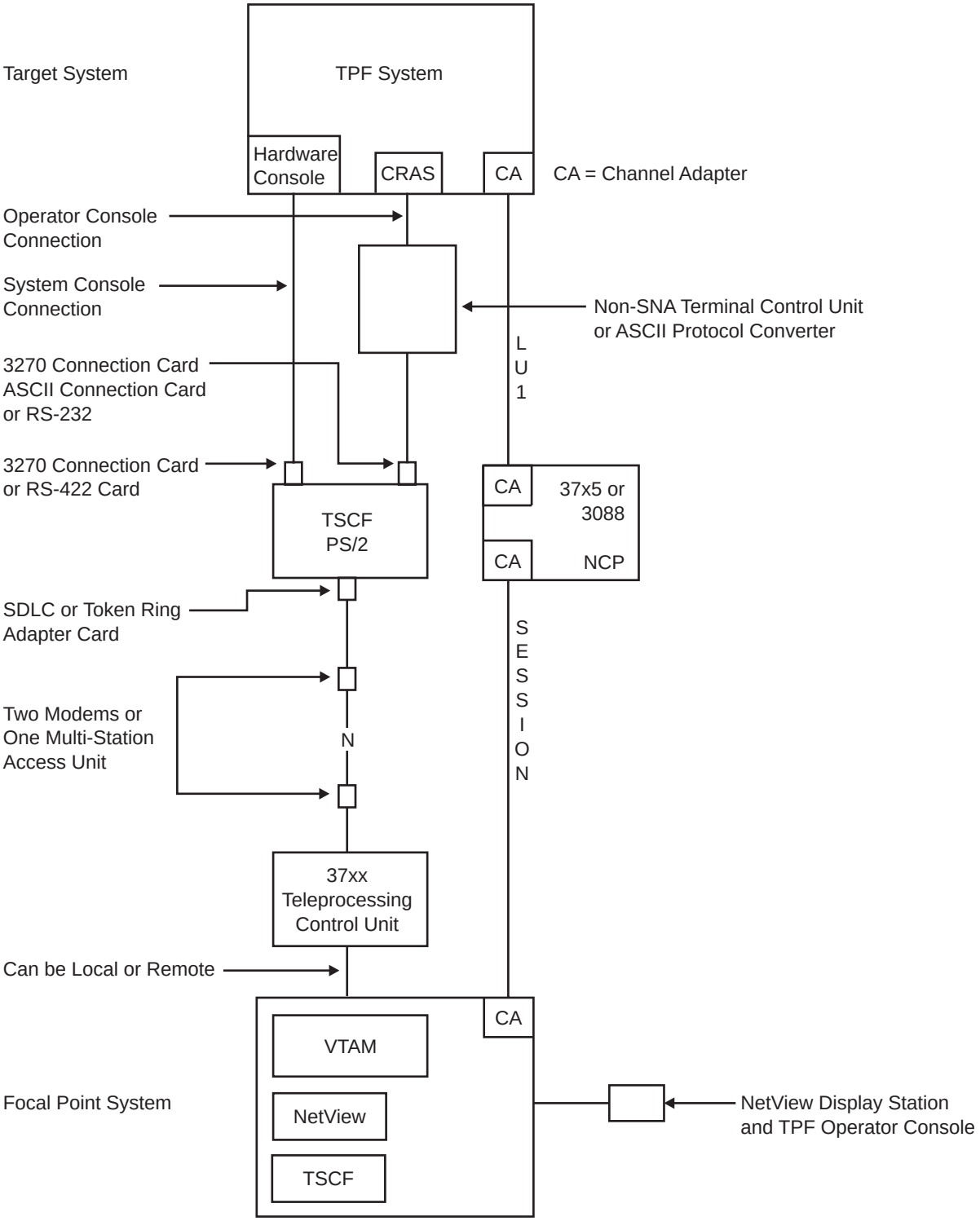



*Figure 1. Sample Hardware and Software Requirements for Implementing Automated Operations using TSCF and NetView. A token ring LAN may be used instead of the SDLC link.*

For detailed information about PS/2 hardware requirements, see *Target System Control Facility General Information*, GC28-1063 and *Target System Control Facility Planning and Installation*, GC28-1065.

### Terminal Access Facility (TAF)

TAF is a feature of NetView that allows the establishment of LU1 sessions between NetView and the TPF system. The LU1 sessions are used as the TPF system's high speed communication links to NetView. These bi-directional paths carry TPF Automated Operations Enabler message traffic to and from NetView. The TPF system should define the TAF LU1 session as a remote SNA functional support console and send selected messages to NetView across this session. Using NetView allows you to write command lists that aid a operator in controlling a local or remote computer complex.

### Terminal Access Facility Support

To prevent flooding of a PS/2 computer by too much output from the TPF system, a high speed communications link to NetView is provided via TAF LU1 session establishment. This permits the TPF system to send selected output messages to NetView across the TAF session for processing by NetView command lists/command processors. Because the TPF systemaccepts input from an LU_1 (printer-type device), NetView can send commands to the TPF system in response to messages interpreted by the NetView command lists/command processors.

In addition, the TPF system accepts a SNA SIGNAL command on this interface. This enables the NetView operator to clear their terminal if it hangs awaiting a response. NetView operators can send an SNA SIGNAL by entering the command:

**SENDSESS** *SessionID* **,***

where: *SessionID* is the name of the TAF resource in session with the TPF system, and, ***** indicates request permission to send again.

**Note:** The TAF LU1 resource must be defined as a 3287/9 printer to the TPF system via the TPF offline SNA table generation (OSTG) program. In NetView, this resource is defined as a 3767 printer in session with the TPF systemF.

### User Responsibilities

Data integrity must be maintained during user exit processing. Inclusion of this support in the TPF system has minimal impact upon system performance. However, when control is passed to the user exit, it is the user's responsibility to address any performance impact on the system.

You will need to create automation procedures and a message automation table to provide for automation.

## Requirements for Implementing Automated Operations within the TPF System

See Figure 1 on page 4 for an illustration of the hardware and software requirements for implementing automated operations using TSCF and NetView.

## Some Important Things to be Aware of

When referring to Figure 1 on page 4, here are some important things to keep in mind:

- TSCF is required for TPF automated operations when external automation is required for IPL and recovery of the TPF system (non-steady state).
- For native console, the operator console connections for TSCF TPF targets must be defined as 3278 Model 2 consoles.

- Ensure that you have the correct TSCF software requirements for the TPF system. For detailed information on these requirements, see *Target System Control Facility General Information*, GC28-1063.
- To automate the IPL of a TPF system, define the PS/2 computer as the prime CRAS. If you want to define a second TSCF PS/2 for fallback, define it as the alternate prime CRAS. This enables automated operations to IPL the complex, establish the TAF LU1 sessions, and later define these sessions as remote functional support consoles (FSCs) to the TPF system.

To aid in the migration to TPF automated operations, you may choose to define the PS/2 computer as an FSC. This gives you the ability to route messages of your choosing to TSCF running in NetView. A restriction on this is the fact that a PS/2 computer must be locally attached to the TPF system.

In a loosely-coupled environment, a local channel attached FSC cannot be shared by the non-Emulator Program (EP) processors in the complex. Because a 3270 local device is attached to the EP processor in the loosely-coupled complex, the PS/2 computer, when defined as the FSC CRAS, can only be used with the EP processor. This restriction is an impact on loosely-coupled environments. IBM's recommendation is to use TAF FSCs over NCP or SNA-CTC and only use the PS/2 computer for prime CRAS automation.

## Primary Areas of the TPF System Affected by Automated Operations

### Filtering TPF Output Messages

Because too many TPF output messages may flood the outboard automation system, it may be necessary to filter these messages. Filtering is accomplished in a customer-replaceable program that can:

- Completely suppress a message
- Send the message without any change
- Reformat the message before routing it to either *all recipients*, or to *only one FSC recipient* that you may specify.

**Note:** Because the NetView message automation table works off a unique message identifier, TPF messages that conform to standards are easier to automate. Non-standard messages require either reformatting in the TPF system or logic tests in the NetView command list.

### How is This Filtering Done?

Two user exits in the TPF system are provided to facilitate the necessary message suppression, routing, and reformatting. This support is performed in the CRAS output module CVIQ.

In segment CVIQ, an ENTRC call is made to these customer replaceable modules: segment UOP1 and segment UOP2. It is in these segments that customer written code may be added for message suppression, routing and reformatting.

Refer to the *TPF System Installation Support Reference* for a detailed description of the user exits interface.

### Functional Support Consoles

The TPF system permits functional support consoles (FSCs) to be printers or 3278 CRTs (because PS/2s emulate only this device). TPF message filtering, as previously described, should be used to route selected messages to NetView that

require immediate attention. Defining the NetView TAF LU1, or even a PS/2 as an FSC to the TPF system, facilitates this routing of messages to NetView by enabling selection of the FSC as the destination.

**Notes:**

1. FSC printer support is still available. However, only one FSC per FSC type may be defined. For example, you cannot have a 3278 CRT defined as the communications FSC and also have a printer defined as the communications FSC. However, a second communications console type may be defined; see *TPF Data Communications Services Reference* for more information.
2. When an FSC is defined as a CRT, there is no longer any hardcopy printout available for this FSC type.

## TSCF Installation Planning

Within your data processing environment, planning must be done for TSCF requirements, such as hardware, software, migration, and customization. For detailed information on these hardware, software, and migration requirements, see *Target System Control Facility General Information*, GC28-1063. For more information on customization, see *Target System Control Facility Customization and Programming*, GC28-1066. For detailed information on planning and installing TSCF, see *Target System Control Facility Planning and Installation*, GC28-1065.

## NetView Requirements

For details on NetView and VTAM definitions, see *Target System Control Facility Planning and Installation*, GC28-1065.

## NetView TAF Definitions

Included here are some examples of TPF OSTG definitions for TAF terminals.

```
*
* THESE TERMINALS HAVE BEEN DEFINED FOR TAF WHICH RUNS UNDER NETVIEW
*
TAF01O00 RSC  LUTYPE=3289,CCTYPE=3274,MODE=NETVIEW,IATA=0216,SCSBUF=4K
TAF01O01 RSC  LUTYPE=3287,CCTYPE=3276,MODE=NETVIEW,IATA=0217,SCSBUF=2K
TAF01O02 RSC  LUTYPE=3289,CCTYPE=3276,MODE=NETVIEW,IATA=0218,SCSBUF=4K
TAF01O03 RSC  LUTYPE=ANY,CCTYPE=3276,IATA=0315
TAF01O04 RSC  LUTYPE=3287,CCTYPE=3274,MODE=NETVIEW,IATA=0316,SCSBUF=2K
```

Refer to *NetView Operation* for details on defining TAF LU1 terminals in NetView.

**Notes:**

1. All TAF LUs to be used with the TPF system must be defined in the TPF OSTG RSC statements. For detailed information, see the *TPF ACF/SNA Network Generation*.
2. All TAF LUs that have been given an IATA must also have corresponding definitions in the UAT/WGTA. Terminal types TPLU1M1 and TPLU1M2 must be used for TAF terminals with SCS buffer sizes of 2K and 4K, respectively, when coding their UA1TTP/WG0TTP terminal type definitions.

## NetView Command Lists

For details on NetView command lists, see *NetView Command Lists* and *Automated Operations - Using Netview Command Lists*.

# Communications Controller (3705) Support

This function provides the support required to load and/or dump a 3705 Communications Controller from a host system operating under TPF control. Support is provided for 3705-I CUs with type 1 or 2 channel adapters, or 3705-II CUs with type 1, 2, 3 or 4 channel adapters attached to the byte multiplexer channel.

TPF Network Extension Facility (NEF) support depends on either:

- The Network Extension Facility (NEF2 PRPQ P85025), or
- Airlines Line Control Interface (ALCI) feature of ACF/NCP.

All subsequent occurrences of NEF in the text of this book are superseded by the previous information.

The NCP/ACF support is required to support TPF/ACF functions.

TPF Network Control Program (NCP) support depends on:

- Subarea support (T5), which requires SNA Network Interconnection (SNI)
- T2.1 low-entry networking (LEN) support, which requires NCP Version 4 Release 3, or later
- T2.1 Advanced Peer-to-Peer Networking (APPN) support, which requires NCP Version 6 Release 2, or later.

All subsequent occurrences of NCP in the text of this book are superseded by the previous information, unless otherwise specified.

## General Description

### Purpose

In order to execute load/dump in a TPF environment, the necessary load modules and control modules must be brought into the TPF system files. This is accomplished by creating a tape file (load file) of the required modules in a MVS environment. This load file is then accessed under TPF control and the data is stored on the TPF system files. Once this has been done, the TPF system can honor requests (user or control unit initiated) to load or dump a 3705.

When the contents of a 3705 are dumped, they can be examined by first requesting that a tape file (dump file) be created under TPF control, and then requesting that the dump file be processed in a MVS environment in order to obtain a formatted printout of the dump.

### Functions

1. To write to a tape file (load file) all the modules required to carry out the system support function under the TPF system. These modules are: 3705 control modules (load/dump programs) and the 3705 load modules (EP/PEP programs). This function is executed under MVS control.
2. To place the contents of the load file on the TPF system files when requested.
3. To load a specific load module or version to a given 3705 control unit on request (either by command or by 3705 initiated interrupt).

4. To dump the contents of a given 3705 control unit to the TPF system on request (either by command or by 3705 initiated interrupt).
5. To purge accumulated 3705 dumps from TPF files to a tape file (dump file) on request. This function is initiated automatically if a dump is initiated and GET File Storage (GFS) is inactive.
6. To provide status and control information to the user in order to insure adequate control over load/dump.
7. To format and print selected contents of the dump file. This is accomplished under MVS control.

## Relationship to the TPF System

Most functions in this program package reside on the TPF system and operate under its control. The SNA Data Communications function interfaces with this program in order to use the functions it provides. However, several functions of this program operate under control of a MVS system and have some dependencies that must be satisfied. These are described in “Programming Systems” on page 13.

# Data Base and/or Input/Output Description

## Data Base Requirements

This program uses fixed file records to maintain load module images on the system. The size and description is in the 3705 Keypoint Record (produced by SIP macro SKCC0C). Large, long-term file pool records are used to maintain images of the dump records. The description of the record type is in the 3705 Dump Record (CD0DC). Tape files are used as the vehicle for carrying data between the TPF system and MVS; these files are in the 3705 Dump File and 3705 Load File (both produced by SIP macro SKCC0C).

## Input

All functions are designed to be initiated by a command. In addition, the load and dump of the 3705 Controller can be initiated by a system restart or by an interrupt from the 3705. This interrupt can be generated by the 3705 Control Program or by pressing the load button on the 3705 Controller. The input message specifies the function to be activated and, depending on the function, the 3705 control unit address and/or the load module name to be acted upon.

Procedurally, a load module must be placed on the TPF files before it can be loaded into a 3705 Controller. Therefore, an interrupt load should never occur unless a load module is maintained for it. A 3705 Controller can be placed online with a load module already loaded into it. The initial load to the TPF files must contain the 3705 resident support (load and dump) programs. Subsequent loads may or may not contain these programs.

## Output

Various messages issued by each phase of this function will indicate the status of that phase; that is, successful completion, processing error, missing input parameter, etc. The output of the dump-purge function is a dump tape which is used as input to an offline routine to format and print the dump(s).

# Program Organization

## Message Editor

This program consists of two segments, CCLA and CCLH.

CCLA is activated by the System Message Processor, 03-CSMP, upon recognition of the characters `Z3705`. This segment validates the input message and activates the appropriate routine.

CCLH is activated by program CCLA when a dump or reinitialization request is received. This segment will validate the input and then activate the appropriate routine.

## Display/Reinitialize Status

This program consists of one segment, CCLB, with an additional transfer vector, CCL1.

CCLB is activated by CCLA when a display request is received, or by CCLD after a load file is processed. This segment will output the current status of each 3705 on the TPF system as to whether or not a load module has been assigned, auto-dump is inhibited, or a load/dump is in progress. Additionally, a list of all load modules on file is given.

CCL1 is activated by CCLH when a reinitialize status request is received.

## Common Functions

This program consists of one segment, CCLC, with an additional transfer vector, CCLR. CCLC is a utility segment which handles the retrieval, unhooking, rehooking, and filing of the 3705 Keypoint Record.

## TPF Load

This program consists of two segments, CCLD and CCLE.

Segment CCLE contains an additional transfer vector, CCLF.

CCLD is activated by CCLA when a load request is received. This segment opens the load file, reads it, and writes the data to the TPF files. When CCLD reaches the end of a load module on the load file, it passes control to CCLF to handle it. When end-of-file or an error condition occurs, control is given to CCLE.

CCLE handles all the error processing for the load and closes the load file when the load is complete. Error messages are issued via the General Purpose Output Message Writer, 03-CZMM.

CCLF handles processing whenever CCLD deciphers the end of a load module. CCLF gets control in order to update the status in the 3705 Keypoint.

## Load Module Scratch

This program consists of one segment, CCLG.

CCLG is activated by CCLA when a scratch request is received. This segment deletes all 3705 Keypoint Record references to the specified load module(s).

## 3705 IPL

This program consists of three segments, CCLI, CCL2 and CCLJ. Segment CCLI contains an additional transfer vector, CCLK.

CCLI is activated by CCLA when an IPL request is received. This segment determines which load module is to be loaded, and whether or not the control unit is active. It passes control to CCLJ to do the actual transfer of the load module.

CCLK is activated by the dump programs on an auto-IPL. This transfer vector picks up the load module associated with the control unit being IPLed from the 3705 Keypoint, determines if the control unit is active, and passes control to CCLJ to do the actual transfer of the load module.

CCL2 handles all error processing for the IPL program.

CCLJ transfers the load module to the 3705 control unit via CCW level I/O operations. An initial control module is transferred to the 3705. This initial module sets up to receive the second control module. The second control module is transferred to the 3705 and receives control from the first control module. The second control module will then accept the requested load module as it is transferred. This completes the IPL of the 3705 control unit.

## 3705 Dump

This program consists of three segments, CCDA, CCDB, and CCDC. Segment CCDA contains an additional transfer vector, CCD1.

CCDA is activated by CCLH when a dump request is received. This segment initiates the dump of a given 3705 by transferring two dump modules to the 3705 and then requesting that the 3705 local storage information be passed to the host. Control is given to CCDB to handle the remainder of the dump. This segment also updates the 3705 Keypoint to indicate either auto-dump inhibit or auto-dump restore.

CCD1 is activated by a control transfer from the 3705 interrupt handling routine when an auto-dump is requested. If auto-dump inhibit is set, control is given to the auto-IPL segment CCLK, and a dump is bypassed. Otherwise, the action is that of CCDA.

CCDB receives control from CCDA. If GFS is inactive, control is given to the purge routine, CCDD; otherwise, it determines if any dump slots are available in the 3705 Keypoint, and controls the transfer of the 3705 dump to TPF pool files. If the dump was 3705 initiated, control is given to the auto-IPL routine, CCLK.

CCDC is a housekeeping segment which maintains the dump slot table in the 3705 Keypoint.

## Purge Dumps

This program consists of two segments, CCDD and CCDE.

CCDD is activated by CCLA when a purge request is received, or by CCDB when a dump request is received and GFS is inactive. This segment opens the dump file, determines the oldest dump slot, and purges all dumps on TPF files to the dump

file. If a request for a direct dump to tape is pending, control is given to CCDE; otherwise, the dump file is closed, a summary message is sent to RO CRAS, and the entry exits.

CCDE handles the dumping of a 3705 directly to the tape file. When complete, an entry to CCLK is created if it was an auto-dump, and control is returned to CCDD to check for more dumps pending.

## Offline Load

This program consists of one CSECT, CCLOAD.

CCLOAD executes under MVS control. It creates the load file that will be input to the TPF system . It takes the specified load modules from the 3705 libraries on the MVS system, strips all extraneous control characters, adds module identifiers, and places them on the load file.

## Print Dump File

This program consists of one CSECT, CCDUMP.

CCDUMP executes under MVS control. It takes the dump file created under TPF control, and presents it, with the correct interface, to the existing 3705 SSP dump format and print routine, IFLDUMP.

## I/O Support Program

This program, CC3705, the TPF CSECT for 3705 I/O Support, handles all direct interface between the Load/Dump Support programs and the 3705 Control Unit via the interface provided by the RDCTC macro at CCW level.

---

# Programming Systems

In order to execute the offline functions of this program, it is necessary to use a MVS system that has the required 3705 Emulator Program (EP) support installed. The MVS requirements are described in the *EP for 3705 Generation and Utilities Guide*. This is required because the offline load and dump programs interface with the 3705 support modules. A more detailed description is in DCSL-CCS.

# CRAS Support

This document describes the functions and capabilities of the computer room agent set (CRAS) support package. Procedures required to initially configure, modify, and use the CRAS support are also outlined.

Computer room agent set (CRAS) is the name given to devices which have been assigned to control the operation of the TPF system. CRAS devices are used by the system operator to monitor system activity, respond to messages generated internally by the system, perform supervisory functions, and to collect hardcopy output of system activity. The number of devices that may be defined as CRAS is installation dependent (maximum of 115) and can be modified dynamically during online operation. The current configuration of CRAS devices is maintained in the CRAS status table (CR0AT).

There are three classes of CRAS devices. They are:

| | |
|---|---|
| **Prime CRAS (PRC):** | System console |
| **Receive Only (RO) CRAS:** | System RO console (hardcopy of Prime CRAS) |
| **Alternate CRAS:** | Any device in the CRAS status table which is not being used as the Prime/RO CRAS |

The terms *alternate Prime CRAS*, *alternate 1052*, and *alternate console* (henceforth referred to as *alternate console*) describe the devices assigned at system generation time which are reserved for use by the console fallback routines (03-CVKM, 03-CIOLC) in the event of a failure on a system console (PRC or RO). These terms should not be confused with the term *alternate CRAS.* Alternate CRAS devices appear in the CRAS status table; alternate consoles do not.

Additionally, any hardcopy or 3270 local CRT alternate CRAS device logged to *SMP* is eligible for use as a functional support console (FSC). FSCs are defined via the ZACRS command, and typically serve as hardcopy logs of system activity which has been separated by function. For example, definition of a communications FSC will provide a hardcopy log of all communications-related input and output. A functional support console (FSC) can be defined as a 3270 local (CRT) device. When a FSC is defined as a CRT, there is no longer any hardcopy available for this FSC.

**Note:** Although the capability exists for remote 3270 CRTs to be defined as FSCs, the TPF system does not support this environment because of long message processing (scrolling package) requirements.

## General Description

## Device Support

The following devices are eligible for use as CRAS devices:

| | |
|---|---|
| **1052/3215** | Printer Keyboard |
| **1053/1980** | Printer (Output Only) |
| **PS/2** | Installed with the IBM Extended Operations Console Facility/2 (EOCF/2) |
| **1977** | Typewriter Terminal |

| | |
|---|---|
| **4505/2915** | CRT Terminal |
| **3270** | CRT Terminal (Local, BSC, and SDLC) |
| **3284/3286** | Printer (Output Only) (Local, BSC, and SDLC) |
| **3278** | CRT Terminal (Local, SDLC).<br><br>**Note:** To use this terminal as a local device, define it as a 3277. |
| **3287/3289** | 3287/3289 Printers (SDLC) |

**Note:** For detailed information on EOCF/2, see the *IBM Extended Operations Console Facility/2 System Administrator's Guide*.

## System Console Support

The terms *console* and *system console* generically refer to the device designated as Prime CRAS at system generation time. Similarly, the terms *system RO*, *system printer*, *RO console*, and *system RO console* refer to the device designated as RO CRAS at system generation time. Hence, console support pertains to the TPF functions which support the use and continued operation of the system consoles. The system consoles, as they are defined via the system generation process (SIP), are assigned symbolic line numbers 00 and/or 01. In a system generated with 1052 console support, both the system console and the RO console are assigned symbolic line number 01. In a system generated with 3270 native console support, the system console is assigned symbolic line number 01 and the system RO is assigned symbolic line number 00. Although the system console devices may be replaced (ZACRS REP) with alternate CRAS terminals during online operation, the replacement devices will not be assigned the system console symbolic line numbers of 00 and 01; they will retain the symbolic line numbers they were assigned at generation time.

There are two mutually exclusive versions of console support: 1052/3215 console support and 3270 native console support. These are the only device types currently supported as system consoles (that is, on symbolic lines 00 and 01). The type of console support is specified by the user at system generation time via the SIP CRASTB macro. See *TPF System Generation* for additional information about CRASTB.

### 1052/3215 Console Support

When the system is generated with 1052 console support, a 1052/3215 printer keyboard or equivalent serves as the system console.

With this support, the prime CRAS and RO are most often defined as the same physical device, with symbolic LNIATA 010000 assigned to the PRC and symbolic LNIATA 010002 assigned to the RO. Defining the PRC and the RO as the same physical device means that one hardware subchannel address is used to access the device. This allows the use of a single symbolic line number (01) for both the PRC and RO. Alternatively, the RO can be defined as a separate device. See *TPF System Generation* for detailed information about the SIP CRASTB macro.

The user also has the option of defining an alternate 1052 to be used for fallback purposes.

Refer to the following table for a description of various fallback scenarios:

## 3270 Native Console Support

When the system is generated with native console support, locally attached 3270 devices (including the native console, if one exists) serve as system consoles. With this support, the PRC and RO are *never* the same physical device; the PRC has symbolic LNIATA 010000 and the RO has symbolic LNIATA 000000. Since the PRC and RO are different physical devices, they are accessed via different hardware subchannel addresses, and therefore require different symbolic line numbers. Up to seven alternate system consoles and seven alternate RO consoles may be defined. See SIP CRASTB macro in *TPF System Generation*. Both the PRC and RO are supported as model 2 devices (1920 byte buffer), regardless of actual device characteristics, as are all 3270 local devices logged to SMP.

Since the system console is not a hardcopy device, all messages destined for it are duplicated on the system RO, providing the RO is valid. If at any time, the RO becomes invalid and there is no useable fallback RO device, message duplication is bypassed. Messages destined for the RO are not duplicated on the PRC.

The TPF system can survive below 1052 state without an RO. If there is no RO available during restart, the TPF system will not attempt to send any messages to it. Below 1052 state, if an RO is not available, any message destined for the RO will be sent to the PRC instead. However, once 1052 state is reached, all messages destined for the invalid RO will result in CTL-86 system error dumps (invalid line number). At this time, the RO should be replaced with another valid device. See the ZACRS command in *TPF Operations*.

The 3270 system console screen is formatted in a manner similar (but ***not*** identical) to that of CMS running under VM. In particular;

- Rows 1 to 21 are reserved for output (protected).
- Rows 22 and 23 are reserved for input (unprotected).
- Row 24 is reserved for unsolicited message notification and other system messages (protected).
- The cursor is normally positioned at row 22, column 2.
- When the screen fills (rows 1–21), `MORE...` will be displayed at the bottom right hand corner, and the screen may be cleared by hitting CLEAR, PA2 or ENTER.
- At SIP time, the user may specify a `MORE...` timeout value, or use the default of 30 seconds. If `MORE...` is on the screen and the timeout expires, the next message sent will force the screen to be cleared, and queued messages will be displayed.

However,

- Pressing ENTER when `MORE...` is on the screen does not cause the current screen to be held (VM's `HOLDING`).
- The RETRIEVE function is only available for terminals on line 01. Hitting TEST REQ on a 3277 or SYS REQ on a 3278 will only cause prior input messages to be redisplayed if the CRT is on line 01 (is the PRC).
- Pressing PA1 does NOT put the system into `CP READ`, but (as on a local 3270) enters `System Service` mode and allows the console to talk to the Log Processor (03-CLG).

## Selection of the System Console During IPL

When the IPL program finds it necessary to communicate with the system console, it will first use the SIP defined subchannel addresses. See SIP CRASTB macro in *TPF System Generation*. The subchannel address of the first device that IPL can successfully *talk to* is selected as the system console. If none of these devices is

operational, IPL will enter an enabled wait state. An attention interrupt generated from any other configured device will cause IPL to select that device as the system console. All communications tables will be built to reflect the selected device as the system console. If the attention interrupt method of console selection is used, the type of device which generates the interrupt must be compatible with the type of console support that is generated in the system. For 3270 native console support, this device must also be on the console control unit. See “3270 Native Console Support” on page 17. Violation of this constraint will confuse the IPL program, and the IPL will not be successful. For example, on a system generated with 1052 console support, generation of an attention interrupt from a 3270 device in response to IPL’s enabled wait state will cause the IPL to be terminated.

During IPL, the 3270 system console screen is formatted as follows:

- Rows 1 to 24 are reserved for output (protected unless input is requested).
- The cursor position indicates the input area for responses. This area falls on the same line and one position to the right of the last character of the output message requesting the response.
- When the screen is full, the next output message will cause the screen to be cleared, and the message will appear on line 1 (the `MORE...` display is not supported during IPL).

### Selection of the System RO During IPL

After IPL successfully selects a system console, it will attempt to duplicate the message sent to it on the system RO. In selecting the system RO, IPL will first use the SIP defined subchannel addresses (reference SIP CRASTB macro). The subchannel address of the first printer that IPL can successfully *talk to* is selected as the system RO. If none of these devices is operational, the IPL program will send a message to the system console indicating this fact, and the IPL will continue. Message duplication to the RO will therein be bypassed.

### Console Fallback

Two types of console fallback are supported: automatic fallback and operator initiated fallback.

In the event of a hardware failure detected by the control program on either the system console or the RO console, automatic fallback will be initiated. Fallback is first attempted to the devices specified as alternate consoles at system generation time. These devices are reserved for fallback purposes and can not otherwise be used (unless reconfigured onto different subchannels). If fallback to one of these devices is successful, notification will be sent to the new device; if unsuccessful, the CRAS table (CR0AT) will be searched for an eligible fallback device (see 03-CVKM). For a system generated with 1052 console support, an alternate CRAS CRT is eligible for fallback of the system console only if it has an associated printer. If no fallback device is found, the system is not operational and a system switchover must be performed.

Alternatively, the operator may wish to initiate fallback (for example, to take a console device offline for scheduled maintenance). The ZACRS command provides this capability. For 1052 console support, the alternate console must first be validated (via ZACRS) before it is eligible for fallback. See *TPF Operations* for information about the ZACRS command.

## Functional Support Consoles

The system RO provides a hardcopy log of system activity. However, as system activity increases, the RO can become overloaded with the increased message

traffic. Also, with the increased message volume, this system log becomes increasingly more difficult to analyze. Functional support consoles (FSCs) were designed to alleviate these problems. They are typically hardcopy alternate CRAS devices which serve as hardcopy logs of a functional area of system activity, and as such may be viewed as a logical extension of the system RO console. Instead of one hardcopy log of system activity, hardcopy FSCs provide the conceptual equivalent of multiple system ROs, each RO collecting output messages for a separate functional area of the system. 3270 local CRTs can also be defined as FSCs; however, with this definition, no hardcopy logs are available for this FSC type.

All commands are associated (via the command editor tables in 03-CSMP) with one or more functional areas. Each functional area corresponds to an FSC. When a command is input to the system, in addition to sending the response to the originator, copies of the input message and the response are sent to the corresponding FSC(s). Unsolicited messages (those not associated with a command) are also sent to the FSC selected by the message sender. Assignment and deletion of FSCs is operator initiated (ZACRS) and may be modified dynamically as dictated by varying levels of system activity. The system RO is the default for all messages destined for an FSC which is not currently assigned. Duplicate messages will never be sent to a single device.

Sixteen FSCs are supported. Each FSC can be referenced by number (1–16) and/or name (reference macro RTCEQ). Seven FSCs are currently named as follows:

| Number | Name | Description |
|---|---|---|
| 1 | RO | System RO printer |
| 2 | PRC | System console |
| 3 | TAPE | Tape console |
| 4 | DASD | DASD console |
| 5 | COMM | Communications console |
| 6 | AUDT | Audit trail console |
| 7 | RDB | Relational database (TPFAR) messages |

**Note:** Although the capability exists for remote 3270 CRTs to be defined as FSCs, the TPF system does not support this environment because of long message processing (scrolling package) requirements.

The procedure for the addition of a new name to this list is outlined in “Addition Of A New FSC Name” on page 30.

## IBM Extended Operations Console Facility/2

An IBM Extended Operations Console Facility/2 (EOCF/2) configuration has an automation gateway that connects the TPF system with a token-ring local area network (LAN). The automation gateway consists of a communications gateway or bridge that emulates a 3215 console. The attached LAN consists of primary system console, alternate, and various FSC sessions implemented with automation capabilities on workstations. See Figure 2 on page 20 for an overview.

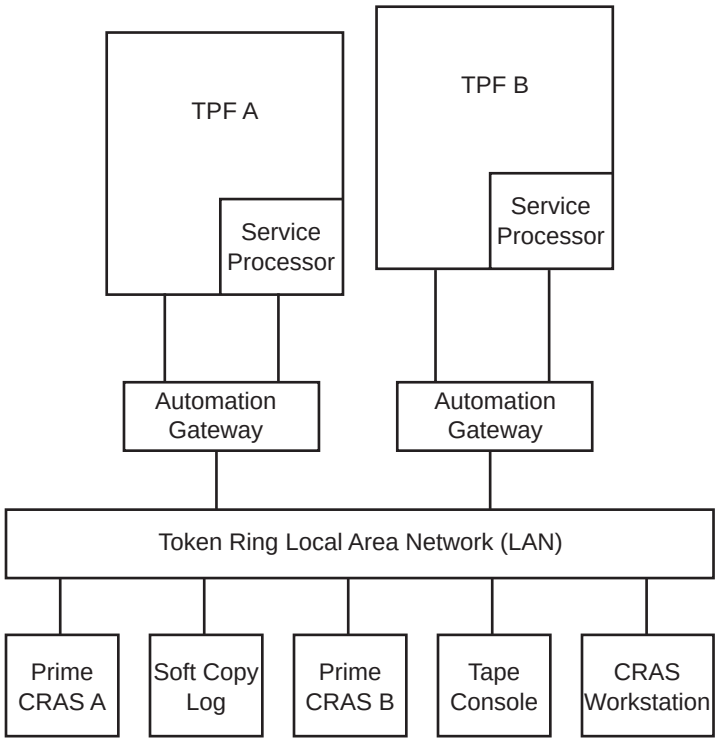



*Figure 2. TPF System Configuration with IBM Extended Operations Console Facility/2*

Except for the Prime and RO CRASs, these LAN CRASs are not defined in the CRAS table. LAN functional consoles receive specific messages because of the functional support console (FSC) routing indicators in an output header that prefixes messages sent to the automation gateway.

## Multiple Console Support

The automation gateway attaches a multiple console support header in front of each message coming from the LAN to TPF, and processes the header for each output message going from the TPF system to the LAN. For multiple console support, individual workstations or windows are identified by a console ID in the multiple console support header in the input and output messages. Each multiple console support header begins with the escape character (, MCSESC in LINEQ) followed by the console ID which is 2 EBCDIC hex digits. The console ID ranges from 00 and 0 through mm, where mm is the maximum terminal address specified in your IBM EOCF/2 configuration. (For additional information on the maximum terminal address, see the *IBM Extended Operations Console Facility/2 System Administrator's Guide*.) Internally in the TPF system, the console ID is stored in hexadecimal, rather than EBCDIC hex digits, in the TA portion of the LNIATA.

You must ensure that all possible LNIATAs on line X'01' are in the WGTA table. See *TPF System Generation* for detailed information about generating additional UAT entries and the creation of the WGTA table. Table 1 contains additional information on the console ID. Figure 3 on page 21 and Figure 4 on page 22 contain examples of multiple console support headers.

*Table 1. Console ID Information*

| FSC Type | Console ID (TA) | Comments |
|---|---|---|
| PRC (Prime CRAS) | 00 (X'00') | If sent to or from the TPF system, this is used to identify the Prime CRAS logical terminal. |

*Table 1. Console ID Information (continued)*

| FSC Type | Console ID (TA) | Comments |
|---|---|---|
| Reserved for future IBM use. | 01 (X'01') | |
| RO (Receive Only CRAS) | 02 (X'02') | • This console ID is considered to be the RO CRAS for this TPF system. This console ID should never be sent by the automation gateway, but, if it is, the TPF system processes it as though it was any other console ID.<br>• The automation gateway considers that the RO CRAS is also the soft copy log. In other words, there is no actual screen or printer designated as RO CRAS. |
| Other CRAS Workstations | 03–mm (X'03'–X'mm'), where mm is the maximum terminal address specified in your IBM Extended Operations Console Facility/2 configuration. | General Use |
| Reserved for future IBM use. | F0–FF (X'F0'–X'FF') | |

**Notes:**

1. If there is no multiple console support input header on a message from LN 01, the message is routed to the Prime CRAS.
2. Except for the Prime and RO CRASs, the LAN CRASs on line number 01 are not defined in the CRAS table.
3. For additional information on the maximum terminal address, see the *IBM Extended Operations Console Facility/2 System Administrator's Guide*.

***Multiple Console Support Header Examples:*** Figure 3 illustrates the format of the input message from an automation gateway to the TPF system. For more information about the maximum terminal address, see the *IBM Extended Operations Console Facility/2 System Administrator's Guide*.

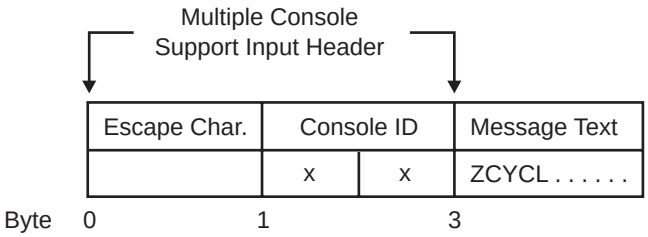



*Figure 3. Example of an Input Message from an Automation Gateway to the TPF System*

Where:

**xx** EBCDIC hexadecimal digits ranging from 00 and 02 through the maximum terminal address specified in your IBM Extended Operations Console Facility/2 configuration.

Figure 4 on page 22 illustrates the format of the output message from the TPF system to an automation gateway. For more information about the maximum terminal address, see the *IBM Extended Operations Console Facility/2 System Administrator's Guide*.

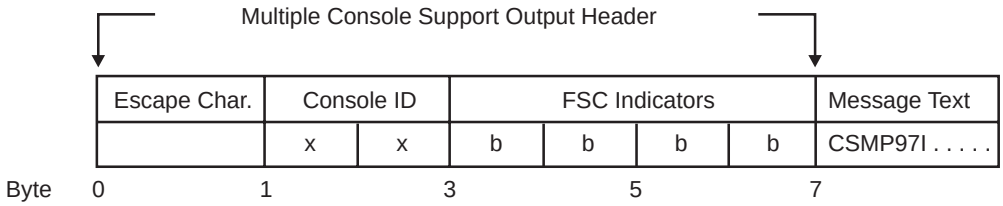



*Figure 4. Example of an Output Message from the TPF System to an Automation Gateway*

Where:

**xx** TA converted to EBCDIC hexadecimal digits ranging from 00 and 02 through the maximum terminal address specified in your IBM Extended Operations Console Facility/2 configuration.

**bbbb**
EBCDIC hexadecimal digits representing functional support console (FSC) indicators used to route messages to specialized consoles. Multiple FSC indicators can be set.

Possible FSC indicators (as defined in RTCEQ) are identified in the following table.

*Table 2. Possible FSC Indicators as Defined in RTCEQ*

| FSC Type | FSC Indicator |
|---|---|
| RO CRAS (Receive Only CRAS) | 8000 |
| PRC (Prime CRAS) | 4000 |
| Tape | 2000 |
| DASD | 1000 |
| Communications | 0800 |
| AUDT (Audit Trail) | 0400 |
| 10 additional user-defined functional support consoles | 0200-0001 |

## Input Message Restrictions

This support allows the operator to control the terminals that are allowed to input critical commands. Commands deemed appropriate for this type of control are defined in different ways and are indicated in the command editor tables (03-CSMP). Modification of these tables is via source update or command. For detailed information, see “Modifying the Command Editor Tables” on page 31. These types of restrictions can be defined for commands:

- The message can only be input by the prime CRAS terminal.
- The message can only be input by authorized CRAS terminals. This type of authorization is referred to as *restricted*. (Detailed information follows.)
- The message can be input by any CRAS terminal.
- The message can be input by any terminal.
- The message itself is only allowed when the system is operating in test mode.

The remainder of the discussion details the restricted type of authorization. With this support, a CRAS terminal must first be granted authorization before it is allowed to input any message classified as restricted. The system operator may grant or rescind authorization to a specific CRAS terminal dynamically during online

operation via the ZACRS command. At system generation time, the system console is authorized to input all restricted messages; authorization is not granted to any other CRAS terminal at this time.

Although message restriction is by message, authorization is by function. The functional classification is equivalent to that of the functional support consoles. Sixteen functional classifications are supported, and once again, these functions may be referred to by number (1–16) and/or by name. The following example may be helpful.

Assume that mounting a tape is a function which is deemed critical to system operation, over which the operator wishes to exercise control. In this example, the ZTMNT command would be defined as restricted in the command editor tables. On the initial IPL of the system, the ZTMNT message would be accepted only from the system console; any attempt to input this message from another CRAS terminal would be rejected. As system activity increases, suppose the system operator becomes very busy and cannot *keep up* with all the tape requests. At that time, the operator could use the ZACRS message to authorize one or more alternate CRAS terminals to input TAPE messages. After so doing, the authorized CRAS terminals would be permitted to input the ZTMNT message as well as any other restricted message which is classified as a TAPE message. At any time, if the operator again wishes to obtain exclusive control of tape mounts, he can remove the TAPE authorization from any and all alternate CRAS sets which currently have such authorization.

**Note:** The ZACRS command is critical to this control, and should be defined as restricted. Authorization to input this message should be restricted to the system console (PRC).

## Alteration/Display Of Current CRAS Configuration

The current CRAS configuration is maintained in the CRAS status table (CR0AT) which resides in keypoint record C (CK8KE). CTKC is a core resident keypoint which is shared by all loosely coupled processors in a TPF-generated with the High Performance Option (HPO) feature. The CRAS table may be displayed or altered during online operation via commands. Successful alteration of the CRAS table will be reflected in the DASD copy of CTKC. Notification of this alteration is sent to every processor in the loosely coupled complex. Upon receipt of this notification, each processor refreshes its core resident copy of CTKC with the latest data.

### Displaying the CRAS Table

The current CRAS configuration may be displayed at any time using the ZDCRS command. Displays vary depending on the options chosen on the input message. For more information about the ZDCRS command, see *TPF Operations*. Unoccupied slots are never displayed. FSCs which are not assigned default to the RO and are only displayed if they have been previously assigned and then deleted.

The ZDCRS command provides the following major functions:

- Display the entire CRAS table.
- Display a particular CRAS table entry given an LNIATA or alternate slot number.
- Display the current FSC assignments.
- Display entries of terminals which are authorized to input restricted commands.
- Display the FSC assignments or authorized terminals by processor ID.

### Altering the CRAS Table

The current CRAS configuration may be altered at any time using the ZACRS command. For more information about the ZACRS command, see *TPF Operations*. ZACRS provides the following major functions:

- Add an alternate CRAS terminal to the CRAS table.
- Delete an alternate CRAS terminal from the CRAS table.
- Replace the PRC/RO with an existing alternate CRAS terminal.
- Add a CRT as an alternate CRAS while associating it with a printer currently in the CRAS table. Associating a printer with a CRT in this manner makes the CRT/printer pair eligible for replacement/fallback of the 1052 console. This association provides no other function. A CRT which does not have an associated printer is not eligible as a fallback device for the 1052 console.
- Validate the alternate 1052 console (before operator initiated fallback).
- Validate the 3270 PRC/RO symbolic console. This is necessary only prior to replacing a console (ZACRS REP) with the original symbolic LNIATA (010000/000000) after automatic fallback has chosen an alternate CRAS terminal for use as a console (see 03-CVKM).
- Fallback the 1052 system console to the alternate 1052 console.
- Fallback the 3270 native console to any available 3270 alternate console specified at system generation time.
- Fallback the 3270 native console to a 3270 device on a specific alternate subchannel.
- Assign an FSC to a hardcopy or 3270 local CRT alternate CRAS device.
- Delete an FSC from a hardcopy or 3270 local CRT alternate CRAS device; this automatically assigns the RO as the FSC which was deleted.
- Authorize an alternate CRAS terminal to input restricted messages.
- Remove restricted message input authorization from an alternate CRAS terminal.

## Operational Restrictions/Considerations

Care should be taken when modifying the CRAS table; it is assumed that the user will not request a change to the CRAS configuration that will adversely affect the running of the system. The following restrictions and considerations should be well noted:

- Only the 1052/3215 and locally attached 3270 devices are active below CRAS state.
- The PRC and RO may be changed to a device on a line that is valid and started (or pollable) even though the system is in 1052 state. When this is done, the external interrupt button may be used to cycle the system to a higher state (that is, to a state in which the replacement set is active).
- For 1052 console support, XS0AA records must be defined for LNIATAs 010000 and 010002.
- For 3270 native console support, an XS0AA record must be defined for LNIATA 000000 (do *not* assign an XLMA record for 010000).
- The eligibility of a 1980 as a hardcopy FSC and of a 1977 as a CRAS terminal is satisfied only if they are in Long Message Transmitter (LMT) Time Out mode. Otherwise, the output chain in the XLMA record is restricted to one message at a time, and this would not be an acceptable method of operation for CRAS devices. It is the responsibility of the user to set the terminals, as CRAS alternates, in Time Out mode in the XLMA record. The Alter CRAS program does not check this condition.
- The CRAS table remains unchanged across an IPL.

- Any ZACRS function that requires reformatting messages (for example, swinging queues between unlike terminals) should be done in CRAS state or above. If this restriction is violated, the requested function will be performed but the queue will not be swung.
- The PRC and RO devices can only be replaced (ZACRS REP) with a terminal which is currently defined as an alternate CRAS in the CRAS table.
- The *associated printer* concept has been functionally replaced with FSC support. Associating a printer with a CRT makes that CRT/printer pair eligible for replacement of the Prime CRAS (ZACRS REP PRC) or eligible for fallback from a 1052 console. This association has no other effect.
- A 1052 PRC can be replaced with a CRT only if the CRT is associated with a hardcopy device.
- If a printer is deleted from the CRAS table, all references to it (that is, assigned via the PRT- keyword) are also deleted.
- FSC support:
  - An existing FSC assignment can be changed to another FSC via the ZACRS CHG message. A user may wish to do this if an existing FSC goes down. However, the queue for the original FSC is not automatically swung to the new device; it can either be purged (ZPLMT), acquired by another CPU (ZALMT), swung to the RO (ZACRS CHG ... OPT-DEL for the original FSC), or left on queue until the device is fixed and is reassigned as the FSC. The TPF system does not support hardcopy FSC queue swings to 3270 CRT local FSCs.
  - An existing 3270 local CRT FSC assignment may be changed to another terminal via the ZACRS CHG message for the new terminal. A user may wish to do this if an existing 3270 CRT FSC goes down. The TPF system only supports queue swings between like devices (for example, CRT to CRT or printers to printers). Because there is no LMT queue maintained for local 3270 CRTs, a new terminal will be assigned as a FSC, with no queue swing occurring.
  - In a loosely coupled complex, the same FSC may be shared by all processors in the complex. However, only NEF type devices are supported as shared FSCs. Otherwise, a device is eligible for an FSC only if it is owned by the processor for which the FSC assignment is taking place.
  - When a hardcopy FSC is deleted, the entire queue (at the time of the delete) will be swung to the RO of the CPUID which initiated the delete. Therefore, deleting a shared hardcopy FSC will cause all messages on queue, regardless of which processor they are destined for, to be swung to the same RO. No support is provided to selectively swing the queue for a shared FSC based on each message's destination CPUID. Note, however, that the delete is effected only for the processor which initiated the delete. For example, if device Z is the DASD FSC for both processors X and Y, a delete of this FSC for processor X would swing the entire queue to the RO on X. After the delete, DASD messages for processor X would go to its RO, while DASD messages for processor Y would continue to go to device Z.
  - Although the PRC and RO have FSC designations, they cannot be changed via the ZACRS message (ZACRS CHG). However, replacing the PRC/RO with another device (ZACRS REP) will effectively accomplish this function.
  - LMT Timeout Considerations for Functional Consoles

    Long Message Transmission (LMT) services hardcopy devices, such as 3270 printers and ALC terminals (1977, 1980, and so on), which have hardcopy capability. These ALC terminals may be attached via the Network Extension Facility (NEF). LMT timeout, as it is currently implemented, maintains a

counter of the number of timeout intervals (currently defined as 9) which have passed since the last message was sent to a device. This number is the amount of time between activations of the LMT timeout program, XLJJ. When the number of timeout intervals (user specified in bits 4–7 of field XS0OTM in the XS0AA record) has occurred, a timeout is declared and the message is resent. When four consecutive timeouts have occurred without a corresponding `answerback` received by LMT, a transmission failure is declared, the device is set to an invalid state, and the operator is informed of the failure. If the device is an FSC, the operator will receive the following notification: `LMT FAILURE TO FSC ON xxxxxx-QUEUE HELD`

To illustrate the situation, consider an example. A 3270 local printer (3284) is being used as the communications (COMM) console, and some hardware error occurs at the device (for example, the plug is pulled out). Suppose a command to start the network is input from an alternate CRAS CRT. In response to this request, many messages will be sent to the originating CRT screen, perhaps faster than the operator can read all of them. The COMM console (which by definition is a hardcopy device) is therefore examined in order to read all the messages. Because it is not operable, no messages appear. If the XS0OTM value has been coded at 10, LMT will wait 90 (9x10) seconds before declaring a timeout and 4 timeouts (360 seconds or 6 minutes) before issuing the message that a failure has occurred. This time could be crucial if any unusual conditions have occurred which require operator notification.

From the above, it can be seen that the values set in the example may be unacceptable to some users with large networks and high message volumes. However, there are considerations to be taken into account when setting these values. The time interval used by LMT timeout multiplied by the number of intervals (XS0OTM) should be greater than the time it will take to print the longest message this device is expected to receive. For example, if the 3284 described above can receive 4K buffer of data and prints at the rate of 60 characters per second, 4096/60 or about 70 seconds are necessary to print all the data and generate an answerback. Any value of XS0OTM which declares a timeout before this physical print time has elapsed, would result in this message being timed out.

It is suggested that the user set the time interval for devices which will be used as functional consoles according to the individual needs of the installation. The information listed above is intended as an aid to assist the user in understanding the process and helping him/her set the values appropriately. If the user changes any of the values described above, he must be aware that the values in XLJJ for timeout intervals and the number of retries is hard coded and used for *EVERY* terminal LMT supports. Changing a value may seem appropriate for a 3270 local functional console, but may be totally inappropriate for a remote NEF report printer.

- Input message restriction:
  - Input message restriction is functionally driven, based on the FSC routing codes in the command editor tables (03-CSMP). With the multiple data base function (MDBF), these tables are subsystem unique. Therefore, if a message is routed to the basic subsystem (BSS), the BSS command editor table will be used to determine if the message is restricted; the message table of the subsystem in which the message originated is *not* used. To avoid confusion and provide consistency, it is therefore suggested that the classification of restricted messages be the same in all subsystems.
  - Since message restriction is functionally driven by the FSC designations, a message classified as restricted which has no FSC designation will be

rejected from all terminals. To avoid this condition, a message that has no FSC designation should *never* be classified as restricted.

# Data Areas and Input/Output

## Keypoint Record C

Keypoint record (CK8KE) contains system information required by the IPL and restart sequences to bring the system to 1052 state. It is a core resident keypoint which is shared by all processors in a loosely coupled complex.

The information in CTKC which is pertinent to CRAS support includes the CRAS status table (CR0AT), the initial status of each processor's first two entries of the line status table, and in the case of 3270 native console support, the `MORE...` timeout value and a table of subchannel addresses to be used as alternate consoles in the event of failure on a system console. All of this information is generated via the SIP CRASTB macro (for more information about the SIP CRASTB macro, see *TPF System Generation*). The CRAS table is the only portion of this information that may be modified during online operation (see "ZACRS—Alter CRAS Table" on page 28). All other portions remain unchanged throughout the life of the system.

## CRAS Status Table

The CRAS table (CR0AT) is the focal point for TPF's CRAS support. All information required to support an installation's current CRAS configuration is maintained in this table. The CRAS configuration is dynamic; it may be modified via command during online operation as dictated by varying levels of system activity. It will also be modified by the console fallback routines whenever a hardware failure forces an automatic switch of a system console.

The initial CRAS configuration is defined by the user at system generation time via the CRASTB macro in the system initialization package (SIP). The hardware locations (subchannels) of the Prime CRAS, the RO, and all alternate consoles (for fallback) are specified at this time for each processor in the loosely coupled (L/C) complex. SIP uses this information to build the initial CRAS table in keypoint record (CK8KE). The CRAS table remains unchanged across an IPL.

Certain privileged and system oriented messages are restricted to CRAS devices. The CRAS table is used to control entry of these messages into the system; privileged messages entered by non-CRAS devices will be rejected. Input is accepted from all CRAS devices when the system is in or above CRAS state. Below CRAS state, input is accepted only from 1052/3215 and locally attached 3270 CRAS devices.

Only the Prime CRAS and RO CRAS need be dedicated CRAS terminals. Alternate CRAS devices may be used for application requirements. If a fallback condition selects as Prime CRAS or RO CRAS an alternate CRAS terminal currently being used by an application, the system will automatically log the device out of the application and log it into the system message processor (03-CSMP).

## Line Status Table

The line status table contains the current status of the communication network attached to the multiplexor channel via 2703/3705 EP and locally attached 3270 devices. It is primarily used by the CRAS package to determine the physical device addresses to be used for the system consoles, and whether or not a device is

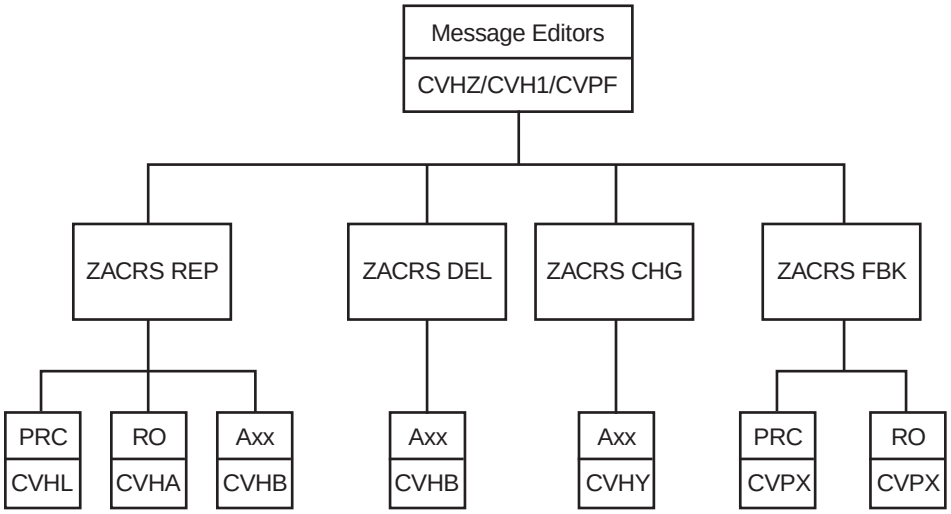
eligible for fallback. To ensure that the system successfully reaches 1052 state, upon every IPL the PRC and RO entries of the LSTB are overlayed with the initial LSTB data (SIP created) from CTKC.

## Command Editor Tables

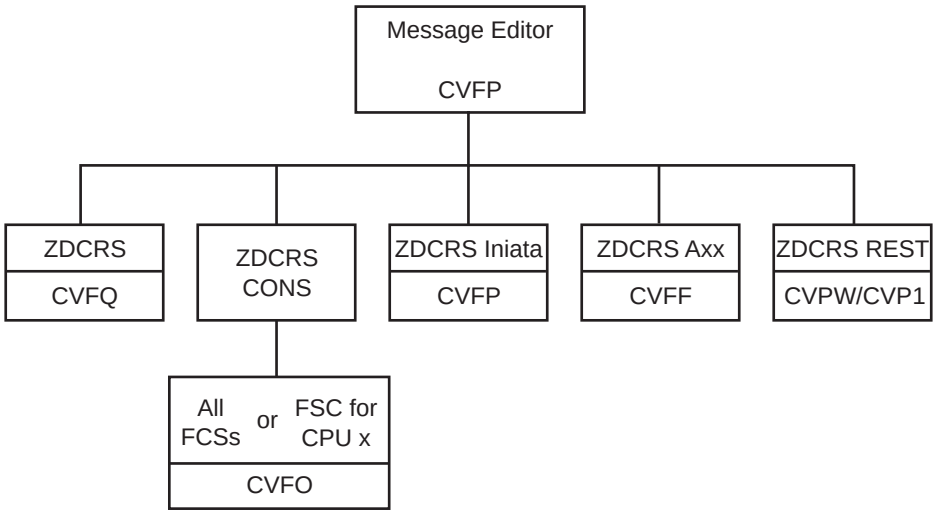
The command editor tables are program and fixed file records that essentially control command input and output; see 03-CSMP for additional information. These tables define each command's input characteristics and indicate which FSC(s) are to receive a copy of the response. See "Input Message Restrictions" on page 22 for further details. Modification of these tables is via source update. For more information, see "Modifying the Command Editor Tables" on page 31 or the ZFMSG command in *TPF Operations*.

# Program Organization

## ZACRS—Alter CRAS Table



## ZDCRS—Display CRAS Table




## Control, Audit, and Reconstruction

The CRAS package is designed to support varying levels of system control based on individual installation requirements. The support is flexible; minimal or maximum control can be exercised based on an installation's specific needs. The level of control is based on how the system is initially configured and installed, but may change during online operation as dictated by system activity. The extent to which the level of control may be changed dynamically is also based on initial system configuration.

System control is provided through the installation's use of the following functions:

- Alternate CRAS definition:

  Commands are accepted into the system only from the system console and from alternate CRAS devices. The user can therefore limit access to the system by limiting the number of terminals defined as alternate CRAS terminals.
- Command indicators:

  Each command carries with it a set of indicators which describes its processing characteristics, many of which pertain to controlling the system in a loosely coupled and/or multiple data base function environment. These characteristics include the following:
  - Message is restricted to the basic subsystem (BSS).
  - Message is restricted to the EP processor.
  - Message is allowed into a dormant subsystem user (SSU).
  - Message is to be transferred to the BSS for processing.
  - Message is to be transferred to the first SSU in this subsystem for processing.
  - Message is restricted to authorized CRAS terminals. (Further details follow).
  - Message is restricted to the prime CRAS terminal.
  - Message is allowed from any terminal.
  - Message is allowed only when the system is operating in test mode.
- Input message restriction:

  Any or all commands may carry the classification of *restricted.* Restricted messages are only accepted into the system from alternate CRAS sets which

have been authorized to do so. The system operator controls which terminals are granted such authorization; he may grant or rescind such authorization dynamically to satisfy varying levels of system activity.

- System activity audit trail:

  Historically, since the system console was usually a 1052/3215 hardcopy device, the RO was used for unsolicited messages and noncritical system messages. With FSC support virtually all system activity is now logged to some hardcopy device, yielding a complete log of system activity. In addition, the user may separate this hardcopy log by functional classification, in any combination he sees fit. The released code supports the following functional categories: PRC, RO, TAPE, DASD, COMM (communications), and AUDT (audit trail). The audit trail has been defined to receive a copy of all output. This provides a chronological log of all system activity. See “Addition Of A New FSC Name” for the procedure to be followed to change or add to these functional categories.

- Reconstruction:

  Every IPL of the system saves the status of the system consoles. For a system generated with 1052 console support, a 1052/3215 device or a 3270 CRT/PRT pair must be available upon every IPL. Alternatively, for a system generated with 3270 native console support, a 3270 local CRT must be available upon every IPL.

  A system IPL can also affect messages destined for FSCs. After an IPL, messages will not be sent to FSCs which are other than 1052 or local 3270 devices until the system is cycled to CRAS state. Such messages will be sent to the RO instead. (Also note that messages on queue for these devices when the system undergoes an IPL will be held until the queue is reactivated in CRAS state.)

# Program Modification Aids

## Addition Of A New FSC Name

Sixteen FSCs are supported, each of which can be referenced by number (1–16) and/or name. Valid FSC names are maintained in the RTCEQ macro. Currently, 7 FSC names are supported: RO, PRC, TAPE, DASD, COMM, AUDT, and RDB. See “Functional Support Consoles” on page 18 for further information. The user can customize this support to fit installation requirements by adding to the list of FSC names. If your system uses an automation gateway, new FSC names may be defined to support additional LAN function console types. To add to the list of FSC names:

1. Update the RTCEQ macro
   a. Add a new RTCDxxxx equate prior to RTCDFSC1, using the *next available bit*.
   b. Add the new console name to the next available &CHECQ slot.
   c. Increase the value of &NCODES by 1.
   d. Increase the value of &COUNTC by 1.
2. Reassemble segments CVFO, CVH1, CVPF, CAPJ, CAPN, CAPP, and CVP1 against the new RTCEQ macro, and reload these segments onto the system.
3. Change the ZFMSG service routine to allow the new FCS to be associated with commands.
   a. Add a line of text to the FSC text table (FSC_TABLE) in the IFMSG macro.
   b. Add a new keyword to the FSC keyword table (CHG_FMT) in the IFMSG macro.

c. Reassemble segments CVAH and CVAI against the changed IFMSF macro and load these segments on the system again.

The new FSC name is now available for use. Note that it is imperative that the new RTCDxxxx equate and the new &CHECQ value occupy the same relative positions.

Example:

To add a new FSC named SERR, update as follows:

- To RTCEQ:

*Table 3. Update to Add a New FSC to RTCEQ*

| | | | |
|---|---|---|---|
| 1. Add new equate | RTCDSERR | EQU | X'0200' |
| 2. Add new name | &CHECQ(7) | SETC | 'SERR' |
| 3. Update &NCODES | &NCODES | SETA | 23 |
| 4. Update &COUNTC | &COUNTC | SETA | 7 |

- To IFMSG:
  1. Add a line of text, **SERR - SYSTEM ERROR CONSOLE**, to FSC_TABLE.
  2. Add a new keyword, **(NO)SER**, to the keyword table CHG_FMT.

After reassembling and loading segments CVAH, CVAI, CVFO, CVH1, CVPF, CAPJ, CAPN, CAPP, and CVP1, the new FSC name is available for use. This means that RTCD-SERR and REST-SERR is accepted by the ZACRS command editor, and that ROUT=SERR is accepted on the WTOPC macro. Also, the new FSC can now be associated with commands through the use of ZFMSG with the **SER** keyword. Note that the same function can be accomplished by using the FSC number (7 in the above example) without any code change. That is, RTCD-7, REST-7, and ROUT=7 are legal values for these parameters.

## Modifying the Command Editor Tables

The command editor tables are program and fixed file records that control the destination of command input and related output. The base definitions, for commands, are contained in program records and consist of the following segments:

**General commands (Zxxxx):** CVAB

**SNA commands (ZNxxx):** CSN0

**Line Control commands (ZLxxx):**
CVLU, CVLV, CVLW, CVLX

**User commands (ZUxxx):** UMET

The base definitions in the released code are generalized in nature. It is suggested that these definitions be studied carefully to ensure they are compatible with specific installation requirements. The tables may be modified when necessary via source update and program reassembly.

Additions and modifications to the base definitions can also be made dynamically during online operation via the ZFMSG command. These changes are maintained in a fixed file record and do not affect the base program records.

Of particular importance is the command input indicators and FSC output routing codes. The command input indicators define the message's processing

characteristics. The user can modify these indicators but must be aware that modification may adversely affect the operation of the system. The FSC output routing codes indicate the FSC(s) to receive a copy of the input message and any associated output. The user can modify these routing codes in any way. For assistance in the initial definition of these tables, see “Installation Information” on page 34.

To modify the base definitions use the following procedure:

1. Code or delete the desired equate for the desired command in the appropriate command editor table. (All equates are defined in the RTCEQ macro.)
2. Reassemble the table and load it onto the system.

**Example**:

The existing entry for the ZACRS command in segment CVAB is coded as follows:

```
DC    AL2(RTCDBSS+RTCDSSD)    Input Indicators
DC    AL2(RTCDCOM+RTCDAUD)    FSC Routing
DC    AL1(PAI_ENTDC)          Activate by ENTDC
DC    AL1(0)                  Additional indicators
DC    C'ACRS'                 Secondary action code
DC    C'CVHZ'                 Name of program to enter
```

This entry specifies the following:

- Input indicators specify that the message is restricted to the BSS (RTCDBSS), and should be transferred to the first SSU in the subsystem for processing (RTCDSSD).
- FSC routing specifies that in addition to responding to the originator of the ZACRS message, a copy of all ZACRS command input and related output is to be sent to the communications console (RTCDCOM) and to the audit trail console (RTCDAUD).
- Segment CVHZ is entered, via ENTDC, to process all ZACRS requests.

To change this entry to define the ZACRS command as a restricted message (RTCDRST), and to assign the PRC FSC routing code (RTCDPRC) to it, the entry should be coded as follows:

```
DC    AL2(RTCDBSS+RTCDSSD+RTCDRST)    Input Indicators
DC    AL2(RTCDCOM+RTCDAUD+RTCDPRC)    FSC Routing
DC    AL1(PAI_ENTDC)                  Activate by ENTDC
DC    AL1(0)                          Additional indicators
DC    C'ACRS'                         Secondary action code
DC    C'CVHZ'                         Name of program to enter
```

The new entry takes effect after reassembling and loading segment CVAB. Since the message is now restricted, only terminals that are authorized to input PRC, COMM, and/or AUDT messages are permitted to input the ZACRS message.

The previously illustrated example can be accomplished online by entering the following command:

```
ZFMSG CHANGE ZACRS RST PRC
```

# System Configuration

## 1052 Console Support

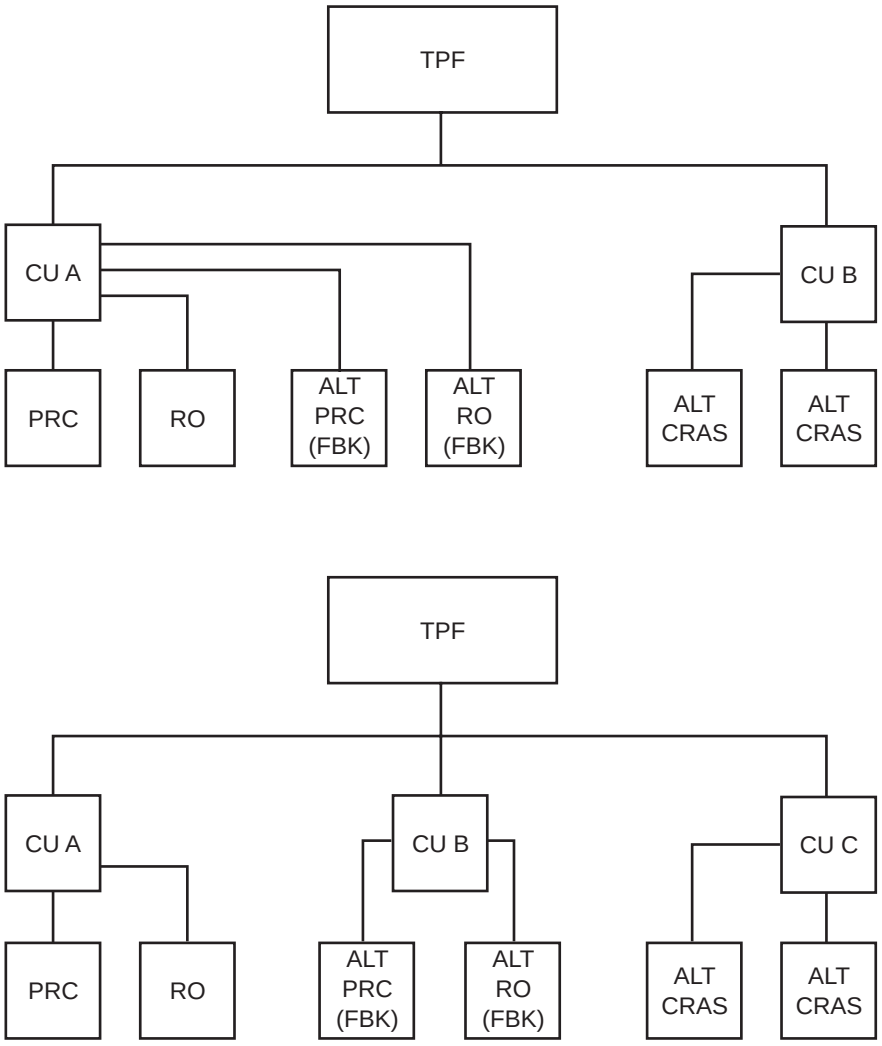
A 1052/3215 printer-keyboard attached through a multiplexor channel is the minimum CRAS configuration for this support. With this configuration, the 1052

must serve as both the PRC and RO system consoles so that operator initiated functions as well as unsolicited logging of output can be serviced on this single device. The operator has the ability at all times to preempt unsolicited output to the 1052 by pressing the request key. This causes an attention interrupt to be stacked in the device. This request is serviced when the output in process is completed, at which time the 1052 is conditioned to receive the input message and forward it to the host processor.

## 3270 Native Console Support

The minimum CRAS configuration for this support is a 3270 console, providing the user does not require a hardcopy log of system activity. Although the TPF system can *survive* without a system RO printer, the user is urged to provide one. The recommended minimal configuration is that a control unit (in addition to that of the native console, if one exists) be dedicated to TPF's system console support. This control unit can be used for the RO console, fallback system consoles, and fallback RO consoles. For information about console definition, see the CRASTB macro in *TPF System Generation*. All such consoles can be on the same control unit or on separate control units, but none may be on the same control unit as any other 3270 local device. All system consoles are supported as model 2 devices (1920 byte buffer), regardless of actual device characteristics.

## 3270 Native Console Support Diagrams




## Installation Information

The code contained in the CRAS package is released with the following characteristics:

- The system console (PRC) on all processors is authorized to input all restricted messages.
- No commands are defined as restricted, although other restrictions may apply.
- Seven of the possible sixteen FSCs are named: RO, PRC, TAPE, DASD, COMM, AUDT, and RDB.
- All messages are assigned a functional FSC designation other than AUDT.
- All messages are assigned the audit trail FSC designation (AUDT).

The degree to which the system can be controlled during online operation is based on the configuration of the command editor tables. For example, if no message is defined as restricted, the authorization function of the ZACRS message is meaningless. Hence, it is suggested that the user carefully analyze these tables to ensure their conformity to installation requirements, and modify them as

appropriate. Procedures for modifying these characteristics to fit specific installation requirements have been outlined elsewhere in this manual.

With the attempt to aid the user in initially configuring the command editor tables to best utilize the support, the following discussion of the philosophy behind the design of the CRAS package is provided. Since the amount of operator involvement required to run a multiprocessor, multisubsystem TPF complex is much greater than in previous TPF systems, more system control messages (that is, commands) are necessary to run the system.

In previous releases, alternate CRAS terminals could be used to divide the workload among multiple operators, each operator having the ability to run the entire system, since all commands were accepted from any CRAS terminal. The current support provides additional ways of dividing the workload among multiple operators, while at the same time allows dynamic modification of the degree of system control. With this support, as system activity increases/decreases the need for more operators, the restricted message support can be used to dynamically modify the number of CRAS terminals which have authorization to input certain restricted messages. This facility still requires one system operator who controls the entire system, granting/rescinding authorization to other auxiliary operators as needed.

Alternatively, the operation of the system can effectively be broken down by function. For example, by defining all TAPE messages as restricted to the TAPE function, authorizing one alternate CRAS CRT to input TAPE messages, and assigning a TAPE FSC for output, the effect is a *TAPE operator* who has ultimate control over all tape activity. Within this scenario, the tape operator's input device is the only device (other than PRC) authorized to input tape messages, and all tape-related output messages will appear on the tape operator's hardcopy output device (the TAPE FSC). If this concept is extended to other functional areas, the effect is to functionalize the system operation. Hence, instead of requiring all operators to know all areas of the system, the system could be run with a TAPE operator, a DASD operator, a COMM operator, etc., requiring each operator to be knowledgeable in only one specific area.

In any case, the decisions for how the system will be run and controlled should be made at system installation time to provide the maximum/minimum required degree of control. It is important that the user understand the installation requirements and the use of this support in order to best define the initial CRAS configuration for his installation. Once the system is operational, the CRAS support is flexible within the confines of its initial definition. An alternative method is to change the definition online.

# Unsolicited Message Processor

This chapter describes, in general, the unsolicited message processor with regard to its relationship in the network. A community of users and the scope of their environment as it relates to the unsolicited message processor are discussed. Two major components of the unsolicited message processor process unsolicited messages in different stages:

1. Initial processing required to queue messages and send notification to the destination terminal.
2. Disposition request processing required to comply with the terminal user's request to display or purge the messages.

## Overview of the Unsolicited Message Processor

The Control Program routing support includes this program package to send unsolicited messages to any non-SNA terminal or SNA logical unit attached to a host TPF system. For purposes of this document, a SNA LU will be either a 3270 SDLC terminal logged on to a non-SNA application or an NEF (terminal attached via the NETWORK EXTENSION FACILITY ) LU. NEF is part of the ACF support which provides connection between 6 bi ALC (Airlines Line Control) terminals and NCP. SNA terminals (such as 3600s) that do not have an LNIATA are specifically excluded from receiving unsolicited messages. This is because they may only log on to SNA applications for which unsolicited messages are not supported. This is because the TPF system is not in control or aware of the screen format and may not send data which would jeopardize the integrity of the application's input message format. Messages originate at the CRAS terminal set or within any processor attached to a TPF network. Processors may consist of SCP, the TPF system, or ACF.

The unsolicited message processor consists of a number of program segments and two data records: unsolicited message directory record (CO1DR) and unsolicited message notification list (CO3NL)).

### Purpose

This package is designed to process CRAS and host processor requests to send an unsolicited message. The requests are identified by information in the input message, a parameter list, and the message header. The routing may be to a specific terminal (single message) or to a group of terminals or SNA logical units (broadcast message).

The terminal types supported by the TPF system and ACF have varying characteristics which require complex formatting and procedural support. This package minimizes the formatting and procedural variances while using the TPF system and ACF resources available with the Message Router.

### Options

User application programs may send an unsolicited message to either a single non-SNA terminal/SNA LU or to all non-SNA terminals/SNA LUs assigned to the originating application. The message may be sent to any terminal type with the exception of R/O (receive only printers) devices.

Messages sent to an R/O device via a ROUTC macro are passed by the Message Router directly to the Long Message Transmitter Program or Output Message Transmission Program in the case of SNA messages.

Unsolicited messages may be sent via the following Z-type messages: ZSNDU and ZSNDA. For more information about the message routing commands, see *TPF Operations*.

# General Description

## Objectives

The unsolicited message processor accepts definite commands from a terminal or application program and delivers the attached text to the designated addressee. Since the request is in the form of a command, in the case of a terminal, it would have been entered by a terminal logged in to the System Message Processor (SMP).

## Functions

This package will validate the entry for syntax and content, then all acceptable messages will be queued to all the specific terminals implied by the addressee (first operand). Terminal(s) will then be notified of the pending unsolicited message and the request for the unsolicited message will be processed; with the delivery of the message the entry will be taken off the queue.

When processing an unsolicited message, a reference item is created and inserted in a directory record (CO1DR). This record is retrieved using the address of the destination terminal to compute the needed ordinal number within the directory's record type. A reference item contains information such as:

1. Message and destination terminal characteristics.
2. Destination address.
3. Origin.
4. Message file address.

Broadcast messages require the creation of reference items for the addressed terminals. Once the unsolicited message directory has been updated, a notification message is sent to the destination terminal to inform the operator of the existence of the unsolicited message. This notification, whose form depends on the terminal characteristics, is immediately sent for all broadcast messages. In order to schedule delivery of these messages, an Unsolicited Notification List (CO3NL) is created containing the addresses of each relevant receiving terminal. This list is processed by a component of the unsolicited message processor.

For single messages, the destination terminal is immediately notified of the existence of an unsolicited message.

After receiving notification of existing unsolicited messages, the terminal operator may either temporarily disregard the notice or immediately request a display of the message by following one of the prescribed procedures.

The appropriate input activates the unsolicited message processor, which retrieves and searches the unsolicited message directory for the proper item based on an originating application and the operator's terminal address. The unsolicited message

is formatted appropriately and sent to the requesting terminal via the ROUTC macro, the Output Edit CRT Driver, or the Output Message Transmission Program.

On a time initiated basis, the policing segment of the unsolicited message processor scans the unsolicited message directory (CO1DR) to again advise terminal users of unsolicited messages queued for the terminal. This policing function may also be initiated by a direct request from the terminal user (ZPUMP).

The initialization of the Prime Directory Records (CO1DR) must be accomplished before the unsolicited message processor is used. This entails the formatting of the records in an acceptable manner for subsequent use by the rest of this package.

## Data Usage and I/O

Two record types are used by UMP: the Unsolicited Message Directory (CO1DR) and the Unsolicited Notification List (CO3NL). The former is a series of fixed records that are shared by all the processors in a loosely coupled (LC) complex and the latter are Pool records.

## File Organization

As presently implemented, there are four primary directory records allocated in the system. Where a heavy use of this package is anticipated, this number may be inadequate. For more information, see “Performance Considerations”.

## Initialization Requirements

Before this function is used, the Directory records must be initialized by using the command provided, ZIUMP. The records are not immediately initialized after the message is issued (in 1052 state). The initialization itself takes place when the system is cycled up to a state with GFS activity (above 1052 state).

## Program Organization

All requests for sending an unsolicited message will be executed in the CPU in which it was entered, for the network immediately attached to that host. The one exception to this is sending a message to a specific LNIATA. Sending a message to an LNIATA has the option of including a CPU identification, which if utilized, routes the message, unaltered, to the CPU indicated. The remote CPU must be a valid host and be able to accept and execute the request.

The queueing of unsolicited messages will be done by each host, upon examining its own immediate environment. The basic queueing logic for the two basic types of messages are: (a) for application messages, the terminal must be logged into the application in question; (b) sending to a vertical address implies that locally attached hardware will be the only terminals eligible for the message. This includes both communication and local channel connected terminals.

## Performance Considerations

For each broadcast message sent by the operator, six notification messages are processed every two seconds. This means that three extra messages are processed every second, tying up three ECBs and three 381-byte blocks. This puts additional strain on the system that should be taken into consideration where the system performance is important. Since unsolicited messages are secondary, broadcast message capability should be used with care.

In addition, each user request for an unsolicited message represents an additional message. During peak periods, this additional load may significantly impact system performance and response time.

## User Information

A user exit has been provided in the terminal selection segment. This is the place to add support such as switchable terminals between different hosts, special log on requirements, etc.

## Restriction

The format of the unsolicited message that is sent from an application program must be the same as if it were entered from a terminal (for example, ZSNDA RES0 ...).

# Unit Record Support

Unit Record Support provides the facility to run jobs which require the use of unit record devices. This package provides routines for unit record device allocation, I/O macro processing, I/O initiation, I/O request queueing and I/O interrupt processing. Secondary functions such as UCS buffer load, FCB load, error logging and utility programs (card to printer, card to tape, etc.) are also provided. The Unit Record Support package provides programs to perform the functions using the following records. The functions are described in “Functions”.

| | |
|---|---|
| **CTKB** | Keypoint Record B. |
| **OB1IR** | OBR/MDR Descriptor Record |
| **OB0IR** | Error Interface Record |
| **UR0IO** | Unit Record I/O DSECT |
| **UR1ST** | Unit Record Status Table |

## Functions

- Prohibit certain unit record jobs to be started when system activity is too high.
- Assign Unit Record devices to requesting application programs.
- Service requests to print a line of data to a specified printer via the PRLNC macro.
- Service requests to read a card from a specified card reader via the RDCDC macro.
- Perform the error recovery procedures for the supported unit record devices.
- Activate unit record jobs via the Unit Record Message Editor
- Facilitate the loading of the Universal Character Set (UCS) buffer via a command.
- Perform the loading of the Form Control Buffer (FCB), for 3211 only, via a command.
- Provide a unit record utility program.

## Devices Supported

The following devices are supported:

- IBM 3211 Printer
- IBM 3505 Card Reader
- IBM 2540 Card Reader
- IBM 1403 Printer
- IBM 4248 Impact Line Printer, Model 2 in 3211 emulation mode

Support is provided for up to 7 printers and 3 card readers. The total sum of these must not exceed 8.

## Macro Support

The nature of unit record operations is basically operator oriented and dependent. Therefore, the design and implementation of unit record macro support within the TPF system has to be geared to operator communications and control. Downgrading of the system when using unit record devices could occur due to

holding core blocks for long periods of time relative to the time required by other I/O devices; therefore, programs using unit record macros should be run only when downgrading of the system can be tolerated. Because of this external system dependence, it is strongly recommended that jobs involving unit record devices only be run when system activity is low. Application programmers should insert Delay and Defer macros in their code when their use of core is at a minimum since this will allow more important programs to more fully utilize the system; thus, periods of minimum core block holding should be designed into the Application Program if at all possible.

To enable the user to write to a printer or read from a card reader the following macros are provided:

**USURC** Assign Unit Record Device Macro. This macro assigns the requested unit record devices to the issuing ECB for its life in the system. The devices will be unassigned when the EXITC macro is issued.

**PRLNC** Print a line Macro. This macro uses a line of data to be printed to the specified printer.

**RDCDC** Read A Card Macro. This macro causes the next card from the specified card reader to be read into core. A card with its first two columns equal to C'/*' signifies the logical end of file. Any subsequent RDCDC macro issued by the same ECB to the same reader will not cause any further cards to be read and the EOF condition is posted in the CE1SUD and the CE1SUG indicators.

**URCTC** U/R Control Macro. This macro causes the TPF system to process a user-defined channel program to a specified device. This is mainly used for the unit record error recovery procedure and other system provided functions such as Universal Character Set (UCS) buffer load. Application programs should avoid using this macro.

Each unit record device in the system will have a three character symbolic name associated with it. For printers the name must be one of those from PR1 to PR7. For card readers, the name must be either CR1, CR2 or CR3. A feature, called the *AVAILABLE* device feature, is also supported. The symbolic names PRA and CRA are associated with the available printer and available card reader respectively. If a job requires the use of a certain device type and yet does not have the requirement of a specific device, the AVAILABLE option can be used in the USURC macro and the subsequent unit record I/O macros. In such a case the system will find, upon the user issuing the USURC macro, the first device of that type which can be assigned and associate it with the issuing ECB.

A Unit Record Utility Program (03-CUAL) is provided to perform the following functions:

- card to print
- card to tape
- tape to print

## Input Action

To activate any job requiring the use of unit record devices, a command of the following format must be entered:

**Command:**

```
ZAURS XXXX ---
```

Where: XXXX = Name of the unit record job to be activated.

The System Message Processor (03-CSMP) recognizes the AURS code and passes control to the Unit Record Message Editor (03-CUAE). There is a high priority job table and a low priority job table within the Unit Record Message Editor. The first table contains jobs that can be activated no matter what the system activity level is; whereas the second table contains jobs that can be activated only if the system activity level test is satisfied. The test is a comparison between the number of currently available 128 byte core blocks and the system defined level of 128 byte core blocks that the CREXC macro checks to see if an entry can be introduced into the system. Upon receiving control, the Unit Record Message Editor searches through its job tables for the unit record job name specified in the input message. If the job name is in the high priority job table, the appropriate program is entered to perform the task. If the job name is found in the low priority job table, the system activity level test mentioned above is performed. If the system activity is low enough to initiate the job, the proper program is activated. Otherwise a message is sent to the originating terminal informing the operator of the high level of system activity and the entry is terminated. The operator should re-issue the command at a later time to get the job started.

## CPU Action

When a Unit Record I/O macro is issued, the Unit Record Support CSECT of the TPF system performs error checking to ensure that the macro entry requirements are met. If the macro is validly issued, the Unit Record Support CSECT queues the I/O request, initiates the I/O operation, processes the subsequent interrupts, resets status indicators at I/O completion and performs error recovery when required. See *TPF General Macros* for a description of the functions performed by the unit record macros.

## Output

The return conditions for various unit record macros are described in *TPF General Macros*. In the error recovery procedures, error logging records are output to the Real Time Tape. For the outputs of various unit record jobs, refer to their respective program documents.

# Application Recovery Support

Application recovery support is an option of the TPF system environment. This package uses the application recovery table, AR0RT, as a vehicle for providing a user (application) with message recovery capability.

The integrity of the application recovery table (ART) is maintained by the system; this package provides the application with the means by which an ART item can be obtained, referenced, keypointed, and deleted. At any point in its processing cycle, an application can save input, output, or user data on file and reference it via its associated ART item, and/or it can save data in the optional data area of the ART item. The maximum block size processed by ART is 1055 bytes. An ART item, once obtained by an application, will be maintained, until deleted by the application. Should application processing be interrupted because of application or system error, this package provides a *timeout* function that reactivates the application using Routing Control data (RC0PL) maintained in the ART item.

This package is a tool that the user (application) can use to supplement existing system message recovery capability and/or to combine message recovery function with data base recovery function. This type of implementation can lead to shortened transaction path length and/or improved data integrity. The application can choose the point of recovery the location and contents of log records, if any, and the tracking (timeout) period. This function is independent of line disciplines, and, thus, can be used with BSC or SNA devices. The user determines the extent of and provides for this recovery capability. The Application Recovery (AR) package provides the interface to a system maintained data area.

**Note:** The application interface program, referenced in the Routing Control Application Table, cannot be corefast.

## Scope

This chapter provides you with an understanding of the externals of the Application Recovery Package and explains, with examples, the user interfaces. This user interface pertains to access to the application recovery table (AR0RT). The function includes obtain, locate, file and delete an ART slot. With the information supplied, the application programmer should be able to code using these application recovery facilities.

## ART User Fields

The user functions supplied by this package are to facilitate access to the application recovery table (AR0RT). This table is to enable the user to control the messages or data that will be recoverable via the timeout function.

The following fields are available to the user via a slot in the ART. A slot, once obtained is unique and saved for the life of the entry. This will be until the user has returned the slot to the available list via a delete request. The user must issue a GLMOD macro prior to modifying a field within the ART slot. This table resides in protected core. Upon completion of the change, a KEYRC macro is issued to reset the protection key. The fields available for user modification are:

**AR0FAW**
This will contain the file address of the message or data handled by the user

application. This file address may be user supplied or supplied by the AR package. This file address cannot reference a block greater than 1055 bytes in size.

**AR0TIM**
A numeric value (maximum = 255) input by the user to activate and control the timeout function. The timeout program, on each activation, decrements this value by 1. A zero result causes the message to be routed back to the origin as timeout has occurred.

**AR0SY1,AR0SY2**
Indicators available for user modification. Bits 0–3 of AR0SY1 are defined as follows:

**AR0SY1=AR0USE**
to indicate slot in use. This is set by the AR package on an obtain request.

**AR0SY1=AR0KYPT**
keypoint request indicator set by the user. Data will be written back to file (keypointed) on the next time-initiated activation.

**AR0SY1=AR0RLSE**
release file indicator for message file address contained in AR0FAW. This is set by the user and results in a release file by the AR package on the next delete ART slot request. This is valid only when the data on file is an input or output message block. If not set, it is the user's responsibility to release any file addresses on completion of processing.

**AR0SY1=AR0INPT**
set by the user and used by the AR package when the release file indicator (AR0SY1=AR0RLSE) is set.

**AR0RPL**
This contains the 16 byte user RCPL (RC0PL). At the time of a request to obtain a slot, the AR package stores the RCPL in this field prior to returning to the user. The first 3-bytes of the RCPL are used by the AR package as a key to identifying the slot requested by the user.

**AR0DTA**
This field is strictly for user convenience. If not necessary, it will have zero length. The maximum may coincide with the maximum data area associated with the RCPL (82 bytes). The number of bytes in this field is defined by the user in the PILOT load data.

Field available for user reference:

**AR0SZE**
One byte field which defines the system generated length of the user data field, AR0DTA. This field is variable. The initialization program stores the system generated size for user use when writing to that area (input via PILOT).

# Application Interface

## Interface Conventions

### Input
The following inputs will be expected by the Application Recovery Package:

```
INPUT:

Register-Conventions =
```

```
R1                Address of Parameter List
R7                Address of Save Area

Save Area Format =

Word              Contents

 0                Not Used:  Reserved for Future Use
 1                Address of Previous Save Area
 2                Address of Next Save Area
 3                Register 0 (R0)
 4                Register 1 (R1)
 5                Register 2 (R2)
 6                Register 3 (R3)
 7                Register 4 (R4)
 8                Register 5 (R5)
 9                Register 6 (R6)

Parameter List Format =
```

Will consist of consecutive fullwords starting on a fullword boundary, each fullword containing an address to be passed. Register 1 (R1) contains the address of the start of the parameter list:

```
Word              Contents

 0                Address of user supplied RCPL (RC0PL)
                                 OR
                  Address of core slot (AR0RT)

 1                Address of input parameter area.

                  This parameter area consists of:

                  Byte 0 - Request Indicator

                      The following equates are used to set the bit
                      request indicators:

                      SET = AR0OBTN   obtain a slot  (via ENTRC ARP1)
                            AR0DLET   delete a slot  (via ENTRC ARP2 or ARP1)
                            AR0FILE   file a message (via ENTRC ARP1)
                            AR0LOCA   locate a slot  (via ENTRC ARP2 or ARP1)

                      The following equate is used to identify the
                      address in word 0:

                      SET = AR0SLOC   core slot

                            if not on RCPL address

                  Byte 1 - Level to File core block from for File Request

                      The following equates are used to indicate
                      the level:

                      SET = D0        for level 0
                            D1        for level 1
                            D2        for level 2
                            D3        for level 3
                            D4        for level 4
                            D5        for level 5
                            D6        for level 6

                      Note:   For the level specified, the file address
                              reference word (CE1FMx) must be zero if
```

```
                              the AR Package is to obtain the file
                              address.  If not zero, it contains the
                              user supplied address.

                    Byte 2-5 - Available for address of ART slot on
                        return from obtain/locate request.  Otherwise
                        ignored.

                    Byte 6-17 - Work area required for delete request
                        only, else ignored.  Used by RLCHA macro (RLCHA).
```

### Output

The following output will be expected from the Application Recovery Package. Return is via BACKC.

```
Registers:

R1                  Address of parameter list
R7                  Address of save area
R14-R15             Contents unknown
All others          Restored prior to return

Parameter List:

Word 0              Unchanged (except on delete request, if core slot
                    address was supplied, it will be zeroed)

                    Input Area referenced by word 1:

Word 1              Byte 0      request indicator is zeroed.

                    Byte 1      level indicator is zeroed.

                    Byte 2-5    ART core slot address (normal return,
                                except on delete request when it is
                                zeroed).

                    Byte 6-17   If provided, no valid user data on
                                return.

ECB Work Area:

CE1FMx              May contain file address (file request only)

EBCM01              Error return indicator

                    (Reference AR0RT, Application Recovery Table, for
                    equates used to identify possible error returns.)

ART Work Area:

AR0SY1-SY2          Indicators are turned off at completion of request.
```

## Examples

### To Obtain A Slot

```
INPUT:

     Parameter List:

       Word 0     Address of user supplied RCPL
       Data Area: Byte 0      AR0OBTN - request to obtain a slot
                  Byte 1      Not used
                  Byte 2-5    Available
OUTPUT:
```

```
     Parameter List:

       Word 0     Unchanged from input
       Data Area: Byte 0      Obtain indicator zeroed
                  Byte 1      Unchanged
                  Byte 2-5    Address of ART core slot for application
                              use.
```

## To File A Message

```
INPUT:

     Parameter List:

       Word 0     Address of ART core slot

                  Note:   If RCPL address is supplied, this program
                          will search ART to locate ART core slot
                          for update.

       Data Area: Byte 0      AR0FILE+AR0SLOC
                              (to file slot and core address supplied
                              in word 0)
                  Byte 1      D4 - data on level 4
                  Byte 2-5    Available

     ECB Work Area:

       CE1FM4     Zeroed

     ART Slot Work Area:

       AR0SY1=
       AR0KYPT    Keypoint request during time-initiated keypoint.

OUTPUT:

     Parameter List:

       Word 0     Unchanged from input
       Data Area: Byte 0      Indicators zeroed
                  Byte 1      Level indicator turned off
                  Byte 2-5    Address of ART core slot

     ECB Work Area:

       CE1FM4     File address

     ART Slot Work Area:

       AR0SY1     Keypoint indicator turned off
       AR0SY1=
       AR0RLSE    AR package supplied file address.
       AR0FAW     File address (same as CE1FM4)
```

## To Locate A Slot

```
INPUT:

     Parameter List:

       Word 0     User supplied RCPL address
       Data Area: Byte 0      AR0LOCA
                  Byte 1      Not used
                  Byte 2-5    Available

OUTPUT:
```

```
     Parameter List:

       Word 0     Unchanged from input
       Data Area: Byte 0      Locate indicator zeroed
                  Byte 1      Unchanged
                  Byte 2-5    Address of ART core slot
```

### To Delete A Slot Reference

```
INPUT:

     Parameter List:

       Word 0     RCPL address
       Data Area: Byte 0      AR0DLET
                  Byte 1      Not used
                  Byte 2-5    Available
                  Byte 6-17   Work area


OUTPUT:

     Parameter List:

       Word 0     RCPL address (if ART core slot supplied:  this
                  field is zeroed).

       Data Area: Byte 0      Delete indicator zeroed (if ART core slot
                              supplied:  core slot indicator is zeroed).
                  Byte 1      Not used
                  Byte 2-5    Zeroed
                  Byte 6-17   No valid data
```

### To Locate A Slot

```
(Previous obtain not done)

INPUT:

     Parameter List:

       Word 0     RCPL address
       Data Area: Byte 0      AR0LOCA
                  Byte 1      Not used
                  Byte 2-5    Available

OUTPUT:

     Parameter List:

       Word 0     Unchanged from input
       Data Area: Byte 0      Zeroed

     ECB Work Area:

       ECBM01=
       AR0NBR4    Slot not found
```

## Utility Programs

Aside from the application functions, this package supplies its own form of maintenance and initialization. These functions are system controlled but the user has the ability to request keypointing.

## Initialization

The first time the application recovery table (ART) is loaded by GOGO (application core load and restart program), the ART initialization program (ARPI) is activated. Using the information supplied in the ART header (user input via a PILOT), each ART slot is initialized and the indicator set to available in preparation for the online application functions. For reactivation of this program due to restart, if the user has requested no keypointing (SIP input via macro MSGRT: KEYPT=NO) the core table is reinitialized to zeroes as specified above. If keypointing is requested, a maintenance function is activated during the cycle to Norm (ARPC). Each slot in-use is checked for a timeout value (AR0TIM=0). If a timeout value is present, the slot is bypassed so that the tracking can continue using the keypointed timing value. The timeout program (ARPT), to be discussed later, will handle this slot and prevent the possibility of a lost slot due to application error. If there is no timeout, a check is made for the presence of a file address (AR0FAW). If there is no file address or, per user request, the file address is not to be released (AR0SY1=AR0RLSE), the field is zeroed and the slot is returned to the list of available slots. Else the file address and any chains are released by this program prior to zeroing the slot and returning it to the available list.

## Timeout

The timeout program (ARPT) will be activated every 5 seconds or at a user specified interval. This is SIP input via macro MSGRT. This program checks the ART slots in-use. If there is no timeout value (AR0TIM=0), the slot is ignored. If a value is found, it is decremented by 1. A zero value at this time causes this program to route the RCPL back to the origin, via CREMC, indicating that this is a returned message (RCPLCTL0=RCPL0RET, for a returned output message and RCPLMSN=RCPL2POS for a possible duplicate input message). The application program will decide on the next course of action. The ART slot may again be located using the 3-byte identifier in the RCPL (Routing Control Parameter List Format, RC0PL).

## Keypoint

The Keypoint program (ARPK) is a time initiated utility that is used by the initialization program if the user has specified keypointing. The application may also request keypointing. A bit is provided in each slot (AROSY1=AR0KYPT) which is checked by the keypoint program. If on, the file record which contains this slot is keypointed. Each slot is checked. The keypoint indicator is turned off and, at completion, this program exits.

## Recoup

Recoup will be updated to chain-chase the file addresses imbedded in the ART core slots in-use.

# Application Recovery Package

This chapter is provided as an aid in the implementation and understanding of the Application Recovery Package.

Items for discussion concern data base initialization necessary to implement application recovery. This will explain in detail the macros and/or data records that have to be initialized to include application recovery to an TPF system and the reason for each. It will also identify areas that should be checked but which may require no change at this time.

## General Implementation Steps

- SIP (System Implementation Package) inputs necessary to generate Application Recovery.
- Check the number of records allocated to #GLOBL in the FCTB (FACE Table, FC0TB).
- Check the amount of main storage in use for Global Area 1.
- Create a PILOT tape including a new GOA record (main storage Allocator Record, INSL-GLBL) and the ART records (Application Recovery Table).

These areas are explained in this chapter.

## Define SIP Parameters

The following parameters are provided as input to the GLOBAL SIP macro. See *TPF System Generation* for more information.

**AMSGRC** YES/NO. This parameter coded with yes will generate the code for the Application Recovery Package.

**RCYTO** This parameter is the time resolution for the periodic maintenance of the Application Recovery Table (ART). The value coded has a range of 1-255 seconds. The default value of 5 means that every 5 seconds a time initiated entry is activated which scans the ART for possible timed out recovery items. This time out parameter is stored in Keypoint Record A.

**ARPKP** This parameter identifies if the ART is to be keypointed on system catastrophic and on a time initiated basis. Coding NO means no keypointing will be done and therefore the contents of the ART will be set to the initial settings on restart. Coding YES means both catastrophic and time initiated keypoints will be taken.

## Determine Global and Related File Modification

The GL0BA macro must include the definition of the main storage pointers for the ART. These pointers are referred to by the Application Recovery Package in order to establish addressability to the ART. Accordingly, the GL0BA macro must be updated to include the ART pointers prior to the execution of SIP Stage II. Secondly, the Main Storage Allocation Record (INSL-GLBL) must contain the initial contents for the ART pointers defined in GL0BA.

## Main Storage

The Main Storage Allocator Record controls the loading of the application fixed main storage area (GLOBAL). Since the ART resides protected main storage area a set of load control entries must be included in the GOA record for GLOBAL Area 1. For each ART record three parameters must be provided to describe the file location of the main storage backup, the size and characteristics of the main storage resident area. Each of these parameters is described below:

**GO1EIX** Index number into GL0BA directory for ART record (the pointer to the ART main storage and file address). The number assigned here should be the next available.

**GO1EON** Set to the next available ordinal number on file for #GLOBL (if more than 1 1055-byte record is needed, the ordinal numbers used shall be in sequence).

**GO1CON** Set to the number of bytes to be loaded into main storage. Whatever value used shall define the first item only. Each additional item will use that value - 16 bytes, to account for the header of each record, after the first, being stripped.

Other indicators in this item will show that it is:

- nonkeypointable
- in the protected area (Global Area 1)
- the prime record has an index entry (in GL0BA) and the header will not be stripped
- each subsequent entry will indicate no index entry and the 16 byte header will be stripped.

Figure 5 in “Examples” on page 56 shows an ART record requiring 4 items defined in the GOA to load an ART main storage table of 4144 bytes. The next available index in GL0BA is 49 (first nonkeypointable index) and the next available ordinal numbers in #GLOBL are 48 thru 51.

The user should also be aware that an increase in main storage usage in the global area may necessitate an increase in the amount of main storage allocated to that area. This value is set in Keypoint Record A via SIP macro GLOBAL. Also the size of the GL0BA directory is defined in the GLOBZ macro. Care should be made that the directory size (in doublewords) has sufficient space for an additional index item.

## Global Records

### GL0BA

```
Name           Number of Bytes        Description
@ARTCR                4               main storage address of ART
@ARTFA                4               File address of ART
```

The above values are set by GOGO (INSLLBL) at restart.

## Files

The Application Recovery Table (ART) is a main storage table with file taken from record type #GLOBL. This may require SIP input modification to the FCTB (Face Table, FC0TB) to increase the number of records allocated to #GLOBL.

## Macros

For possible Phase I and III testing and the PILOT load, DRIL (Data Record Information Library) has been updated to include the ART DSECT (AR0RT).

## PILOT

The PILOT tape contains predefined sets of data records that can be loaded to the online system. The Application Recovery Package uses program segment GOGO of the Application Main Storage Load and Restart Program (INSL-GLBL) to load the ART main storage record and ARPI (the Application Recovery Initialization Program) to initialize it in main storage. Prior to the activation of these programs the file copy for the ART must contain data which will be used by these programs. The PILOT tape is the means of loading that necessary data to the online system.

Three parameters must be initialized in the first ART record. These parameters are described below and define the main storage size of the location of the file copy of the ART.

**AR0BID**
AR – the ID of the ART record.

**AR0TOT**
Number of slots generated. Care should be taken that number will not exceed the amount of main storage reserved for the ART.

**AR0LEN**
Length of variable length of slot (AR0DTA). This up to the user and valid values range from 0 to 82. This coincides with the RCPL (RC0PL) data area. This field is used by the initialization program to initialize the slots.

**AR0ORD**
Starting ordinal number for the file copy of the ART. This will equal the ordinal number for the first item defined in the GOA (field GO1EON in INSL-GLBL). This is used by the Application Recovery keypoint facility to write the main storage copy back to file and program CPSF for catastrophic error recovery.

The ART ID is necessary for all the file copies relating to the ART. The other 3 fields need only be defined in the first ART record on the PILOT. It was previously mentioned that all records after the first have the header stripped prior to loading. This implies that any data (except the ID) contained in subsequent records is lost and therefore not necessary.

The number of ART records which must be built for the pilot tape is a function of the number of ART slots (AR0TOT) and the length of each slot (AR0DTA). This can be calculated in the following way:

```
10 + (24 + AR0LEN) * AR0TOT     ROUND UP
         1024
```

This number of records must be included in the allocation of global records (record type #GLOBL).

PILOT data for the Application Recovery package has now been discussed. Necessary data includes:

- The above mentioned ART data.
- The GOA with new items defining the ART main storage record
- The GL0BA DSECT to add the new index for the ART to the directory.

Figure 6 in “Examples” shows ART data necessary for the PILOT. The example uses the GOA defined in Figure 5. This example gave us an ART main storage of 4144 bytes. Assume that the user wants the maximum data area (AR0LEN=82) and with this information the number of slots generated is 39.

## Examples

```
GO1GO     GSTAR     CREATE,1.
GO1NUM    ENTIT     1-1-X'07'.          LOAD MODE 7
GO1ENT    ENTIT     1-1-X'0017'.        23 ENTRIES
GO1CRE    ENTIT     1-1-X'0005'.        5 4096 BYTE BLKS
GO1CHN    ENTIT     1-1-X'00'.
GO1FCH    ENT       X'0000'.
BSTA06    ENT       (#GLOBL)1.
GO1BID    ENT       GO.
                     |
                     |
                     v
GO1CON    ENTIT     1-21-X'83000000'    LOCATED IN GLOBAL AREA 1,
                                        SIZE, INDEX IN GL0BA
GO1EIX    ENTIT     1-21-X'0031'.       GL0BA INDEX
GO1EID    ENTIT     1-21-AR.            ART ID
GO1EON    ENTIT     1-21-X'00000030'.   STARTING ORDINAL FROM #GLOBL
GO1CON    ENTIT     1-22-X'81010010'.
GO1EIX    ENTIT     1-22-X'0000'.       NO INDEX
GO1EID    ENTIT     1-22-AR.
GO1EON    ENTIT     1-22-X'00000031'.
GO1CON    ENTIT     1-23-X'81010010'.
GO1EIX    ENTIT     1-23-X'0000'.       NO INDEX
GO1EID    ENTIT     1-23-AR.
GO1EON    ENTIT     1-23-X'00000032'.
GO1CON    ENTIT     1-24-X'81010010'.
GO1EIX    ENTIT     1-24-X'0000'.       NO INDEX
GO1EID    ENTIT     1-24-AR.
GO1EON    ENTIT     1-24-X'00000033'.
          GEND
```

*Figure 5. Example of an ART Record to Load an ART Main Storage Table*

```
AR0RT     GSTAR     CREATE,1.
AR0BID    ENT       AR.
BSTA06    ENT       (#GLOBL)48.
AR0TOT    ENT       X'0027'.            # OF SLOTS GEN'D=39
AR0LEN    ENT       X'52'.              VARIABLE LENGTH=82 (max.)
AR0ORD    ENT       X'0030'.            STARTING ORDINAL=48
          GEND
AR0RT     GSTAR     CREATE,3.
AR0BID    REP       AR-1-3.             INIT ART ID IN FILE COPIES
BSTA06    ADD       (#GLOBL)49-1-1-3.
          GEND
```

*Figure 6. Example of ART Data Necessary for the PILOT*

# Mapping Support

This chapter provides a detailed description of the procedures and processes required to implement and use the various functions of the mapping support package. The chapter describes:

- The tasks necessary to implement the mapping support package in a TPF system.
- The processes required to create and maintain the online data base.
- The procedures to generate and load the terminal map records.
- The steps to prepare input.
- The messages printed which allow the user to interpret the results of the map creation.
- The general process flow of the Map File Create and Load functions.
- The following devices are supported by this package:
  - 3277 Model 1
  - 3278 Model 1 (default buffer size)
  - 3277 Model 2
  - 3278 Models 2,3,4f (default buffer sizes)
  - 3284/3286 Model 1
  - 3287/89 Model 1 (default buffer size)
  - 3284/3286 Model 2
  - 3287/89 Models 2,3,4 (default buffer size)
  - 3284 Model 3
  - 1977 Printer WITHOUT Tab Feature
  - 1977 Printer WITH Tab Feature
  - 1980 Model 24 Printer
  - 1980 Model 21 Printer
  - 3278 Models 1 and 2 (alternate buffer sizes)
  - 3287/89 Models 1 and 2 (alternate buffer sizes)
  - 3287/89 Models 1 and 2 SLU Type 1 devices

## Implementation Procedures

This section describes the:

- Specific additions and modifications made to control program functions, macros, global areas, data records, etc.
- The steps the user must perform in order to incorporate the AS Package into a TPF system.
- New segments, macros, and data records comprising the AS Package.

Where modifications to existing TPF functions have been made (Global areas, FACE Table, SYSEQ, Recoup, etc.), the intention was to provide support for the previously mentioned terminal types. These terminal types require twelve new record types for the Terminal Map Record file. These record types were added to SYSEQ and twelve additional files allocated in the FACE Table. A user with fewer device types referencing Mapping Support may desire to delete the extraneous

areas or leave the areas free for possible future device expansion. A section of this chapter describes the required steps to follow for modification of the provided support.

## General Implementation Outline

- Modifications to Global Records
- Modifications to the Allocator Record
- Modifications to the System Equate Macro, SYSEQ
- File Requirements
- Program Segments And Data Record Allocation
- Additional Modifications to Programs, Tables and Macros
- Record Initialization
- Translate Table For 3270 Data Stream Interrogation
- Program Resident Tables And Constants Possibly Requiring Alteration
- Steps to Alter the Terminal Types Defined in the Mapping Support Package

## Global Records

These records have been updated to reflect the new main storage resident Map Sharing Table (AS2MT) and a new 256 character translate table used by the Mapping Support Package. These fields have been provided for in GLOBY.

The following description shows the additions that have been made to the GLOBY macro. Note that the order the fields are to be loaded into main storage must correspond to the index number sequence in the Main Storage Allocator Record (GO1GO).

### GLOBY

New global tags were added to reference 2 global records in main storage:

```
@MAPRCD     Map Sharing Table
@T3270      Translate Table
```

These are initialized by the Application Core Load Program; @T3270 provides the address of a 256 Character 3270 Translate Table. This table is used by the Mapping Support Package to detect 3270 commands and orders in the data streams.

### GLOUC

The following tags were added to the table in GLOUC:

```
@MAPRCD
@T3270
```

These items must correspond to the slot updated in GLOBY and enables the file copy of keypointed records to be updated.

## Main Storage Allocator Record (GO1GO)

New items have been added to the Allocator Record for the main storage resident Map Sharing Table (AS2MT) and the translate table used by the TPF Mapping Support Package to interrogate 3270 data streams. The items were added to immediately follow the last record entry for the protected main storage resident records. Their sequence in the Allocator Record corresponds to the order the fields are to be loaded (i.e., GLOBY). To insure that existing displacements in GLOBY were not changed, the values assigned to the GO1EIX field for the new items were greater than the highest previously assigned GO1EIX. Neither table is keypointed.

FACE ordinal numbers were assigned by referring to the Allocator Record and assigning ordinal numbers higher than the existing GO1EON. The addition of these records might cause Global Area 1 to exceed the space allocated to it. If this occurs, it will be necessary to change the number of 4096 byte blocks specified in the CK1GLBK field of Keypoint A (CK1KE) and perhaps the CK1GLIK field of Keypoint A. If CTKA is updated to reflect an increased number of 4096 byte blocks, this new value must also be incorporated in the GO1CRE so that the two values are equal. GO1CRE contains the number of 4096 byte core blocks required.

The following fields were updated:

**GO1EIX** Index number of entry within GLOBY allocated to this record.

**GO1EON** 03-FACE ordinal number of file copy of this record.

**GO1EID** Basic ID of this main storage resident record.

**GO1CON** Number of double words to be loaded into main storage, and keypoint, index, global area, and header information.

The information contained in the following table constitutes the additions that were made to the Main Storage Allocator record as a part of Mapping Support in the 8.3 Release. In most cases, existing users will have to modify these values for their respective system. The basic procedure outlined above should be adhered to for such modifications.

*Table 4. Fields Added to the Main Storage Allocator Record*

| Table | GO1CON | GO1EIX | GO1EID | GO1EON |
|---|---|---|---|---|
| AS2MT | X'84000000' | X'002E' | MD | X'00000033' |
| Translate table for 3270 | X'20000008' | X'002F' | 2 blanks | X'00000034' |

## Mapping Support System Equates (SYSEQ)

The Terminal Map Record (AS0MP, AS1MP) files are specifically device related. There is one file for each terminal type utilizing this package. See “Additions to the FACE Table” on page 60 for the terminal types supported and added to SYSEQ.

The following equates were added to SYSEQ for 3270 control and order codes:

```
#SF      X'1D'  Start Field
#SBA     X'11'  Set Buffer Address
#IC      X'13'  Insert Cursor
#PT      X'08'  Program Tab
#RA      X'3C'  Repeat to Address
#EUA     X'12'  Erase Unprotected to Address
#EM      X'19'  End of Message Character
```

The following equates were added to SYSEQ for 3270 command codes:

```
#WRT     X'F1'  Write
#EWT     X'F8'  Erase/Write
#CPY     X'F7'  Copy
#EAU     X'6F'  Erase All Unprotected
#EWA     X'7E'  Erase/write Alternate
```

The following equate was added to SYSEQ to provide an end of screen separator character in output data streams to the 3278 and 3277 CRTs.

```
#EOSS   X'11'  End Of Screen Separator
```

# File Requirements

This topic contains the formula needed to compute the storage requirements for data records on file and the additions to the FACE Table to reflect these new data records.

## Allocation

Record allocation must be determined for the Map File Index Table File (AS4MF) and the Terminal Map Record File(s) (AS0MP, AS1MP). One Terminal Map Record File must be allocated for each device type which sends and/or receives formatted or mapped data. For *each* MSP File allocated in the FACE Table (FCTB) one large record must be allocated in the MSI File. Twelve files have been allocated in FCTB; thus, twelve records were allocated in the MSI file. The twelve files represent the device types included in the terminal support of this package. Because individual user requirements vary, these file allocations may require modification. However, the user should consider future terminal and file expansion when establishing his allocations. Existing users, should there not be enough file storage available, must run a data base reorganization.

The following table indicates the number of new large fixed file records needed. When the result after division contains a remainder, round up to the next higher integer.

*Table 5. Large Fixed File Record Allocation*

| Record | Comments | Record Type | Number of Records |
|---|---|---|---|
| AS2MT | File backup for core record | #GLOBL | One |
| AS0MP<br>AS1MP | One file for each device type | #TERnn;<br>nn=01−0D | I+O+2B= number of records to allocate for each file corresponding to a device type. |
| AS4MF | | #ASMSI | One record for each MSP file allocated in FCTB |

**In This Table:**

**I =** Number of input maps per terminal type defined by the user.

**O =** Number of output maps per terminal type defined by the user.

**B =** Number of maps per terminal type which have been defined by the user as **BOTH** (that is, same map used for input and output).

To each computed MSP allocation, the user should add an expansion factor to provide for additional maps.

The formula for the MSP file must be computed for each device type unless every user created map is defined for all terminal types. In this case, the record allocations would be equal.

## Additions to the FACE Table

The file address compute program table (FCTB) provided contains the following record types:

*Table 6. File Address Compute Program Table (FCTB) Record Types*

| Record | Record Type | Record Size | Comments |
|---|---|---|---|
| AS4MF | #ASMSI | Large | |
| MSP | #TER01 | Large | 3277/3278 Mod1 |

*Table 6. File Address Compute Program Table (FCTB) Record Types (continued)*

| Record | Record Type | Record Size | Comments |
|---|---|---|---|
| MSP | #TER02 | Large | 3277/3278 Mod2 |
| MSP | #TER03 | Large | 3284/3286 Mod1 |
| MSP | #TER04 | Large | 3284/3286 Mod2 |
| MSP | #TER08 | Large | 1977 W/O Tab |
| MSP | #TER06 | Large | 1977 With Tab |
| MSP | #TER07 | Large | 1980 Mod 24 |
| MSP | #TER08 | Large | 1980 Mod 21 |
| MSP | #TER09 | Large | 3278 M1 Alternate |
| MSP | #TER0A | Large | SLU Type 1 2K |
| MSP | #TER0B | Large | SLU Type 1 4K |
| MSP | #TER0D | Large | 3287 M1 Alternate |

Any modifications to the record allocations in the FACE Table will require a reassembly of FCTB. If record types are changed or deleted in SYSEQ, FCTB will have to be reassembled.

## Main Storage Resident Tables

Record type #GLOBL was allocated enough space to accommodate the new main storage resident, Map Sharing Table (AS2MT) and the new 3270 translate table. The ordinal numbers are specified in the GO1EON fields of their respective entries in the INSL-GLBL Record.

## Additions To The Face Driver (03-DFAD)

The FACE Driver and Offline Interface Program (03-DFAD) was updated to include the record IDs of the Mapping Support records that reside in fixed file storage. The IDs were defined in the same order as their appropriate record type IDs appear in the System Equate Macro (SYSEQ). The purpose of this offline program is to provide a means of verifying and validating the FACE Table (FCTB) record allocations *prior* to loading the new FCTB into the online system. If the user deviates from the table provided, it is recommended that this program be executed before loading FCTB online. The following is a list of the fixed file records and their IDs as they are currently listed in SYSEQ.

```
Record      ID

MSP         MS
MSP         MS
MSP         MS
MSP         MS
MSP         MS
MSP         MS
MSP         MS
MSP         MS
MSI         MA
MSP         MS
MSP         MS
MSP         MS
MSP         MS
```

### JCL To Run DFAD

```
//DFAD     JOB
//JOBLIB   DD  DSN=ACP.xxxx.xxxx.LK,DISP=SHR
//NAME     EXEC PGM=DFAD
//DFADPRT  DD  SYSOUT=A
/*
```

Where:

```
ACP.xxxx.xxxx.LK = The library to which both FCTB and
                   FACE have been linked.
```

## Program Segments and Data Record Allocation

### PDS Members

The following is a list of program segments comprising the Mapping Support Package that have been allocated in the System Allocator (03-SAL). These programs have been allocated to each subsystem. Each has been allocated as disk resident. However, since usage of this package will vary by user, the user may consider assigning some of the more frequently accessed (ASF) segments to main storage.

```
ASC1          ASF1  ASL1
ASC2          ASF2  ASL2
ASC3          ASF3  ASL3
ASC4          ASF4  ASL4
ASC5          ASF5  ASL5
ASC6          ASF6  ASL6
ASC8          ASF7
ASCA          ASF8
              ASF9
              ASFI
              ASFO
              ASFJ
              ASFP
```

The following offline program segments in the Mapping Support Package are written in BAL. Each must be assembled to the MVS object library using the MVS Assembler and link edited to the load module.

```
ASLOL         ASLPR
ASLOM
```

### Data Records

The following is a list of records and their IDs requiring large (1055) pool addresses. These have already been added to the System Allocator (SAL) and are required for accessing random pool record addresses.

| ID | | Record | Description |
|---|---|---|---|
| MS | MSP | Terminal Map Record | Long term, duplicate |
| MA | MSI | Map File Index Table | Long term, duplicate |
| MB | MSS | Page/Scroll Control Rcd | Short term, nonduplicate |

### System Maps

This section lists and describes all maps defined in the macro library for use by TPF programs. These maps are henceforth called *System Maps*.

The initialization and maintenance of the related physical online map records is the responsibility of the user of the TPF system. The Map File Create And Load Program (03-ASL) is provided for creating and loading of maps to the online files.

***Source:*** Each System Map is defined by a group of source statements catalogued in the TPF Macro Source Statement Library under the *mapname* given in this document. These source statements must be copied or included in the user's Map Source Statement Library for input to the Map File Create And Load Program (03-ASL).

***Processing:*** This offline program (03-ASLOL) will create and load physical maps for each group of source statements defining a map and included in the user's Map Source Statement Library.

***Disposition:*** The System Map definitions are included as part of the TPF system. Once generated, these maps remain constant until a change is made in the source statements in the macro library.

***Contents:***

***Last Line Descriptor For Input/Output:***

**Mapname:** C001

**Description:** This map defines the last line of a display terminal which is reserved for TPF usage. The map is designed to provide the TPF system with a means of displaying a message without overlaying a terminal display area on the screen. The TPF system uses this map for short input or output messages.

Input messages are limited to 37 characters maximum. Output messages are also limited to 37 characters maximum.

***Unsolicited Display Descriptor:***

**Mapname:** C002

**Description:** This map defines a format for displaying the unsolicited messages requiring formatting by the unsolicited message package. These messages include all CRAS and some application-program- initiated requests to send unsolicited messages.

The CRAS requests include single and broadcast messages while application program requests will include broadcast messages only. (Single requests are destined for one terminal only. Broadcast requests are destined for more than one terminal.)

This map allows a maximum of 253 characters to be displayed starting at the first position on a screen. The text is displayed as a continuous stream of characters using up to the maximum of 253 display positions. Unused positions are set to blank characters.

***Last Line Descriptor For Output Only:***

**Mapname:** C003

**Description:** This map defines the last line of a display terminal which is reserved for TPF usage. The map is designed to provide the TPF system with a means of displaying a message without overlaying a terminal display area on the screen and moving the cursor.

The TPF system uses this map for output messages of from 1 to 37 characters maximum.

# Miscellaneous Modifications

This section discusses changes that have been made to several peripheral programs and macros.

## Datas

The new data macros have been added to the DATAS macro to allow programs to call two or more data records while specifying only one base register. The new data macros and their respective designations are:

| Macro Name | Description |
|---|---|
| AS2MT | Map Sharing Table |
| AS4MF | Map File Index Table |
| AS8MB | Page/Scroll Control Record |
| AS0MP | Terminal Map Record-Input |
| AS1MP | Terminal Map Record-Output |

## System Message Processor

Segment CVAB of the System Message Processor was updated to recognize the Z-type command to initiate the Map File Create and Load Program. If mapping is system generated, the following message will generate an ENTNC to ASL3. For a system with the Multiple Data Base Facility (MDBF) installed, the Map File Create and Load is initiated for a particular subsystem by prefixing the message with the appropriate Subsystem User ID.

```
                                ex. 1 zamap

                                ex. 2 ssuname//zamap

WHERE SSUNAME = a 1 to 4 CHARACTER SUBSYSTEM USER ID NAME.
```

## Recoup

The Recoup Package must be made aware of the Mapping Records for chain-chasing of the long term pool records. See *TPF Database Reference* for more information about the inputs needed by the recoup descriptors.

## Global Label Table

The Global Label Table, which is used by the Diagnostic Output Formatter (PDVLDUMP) to label global area fields during a systems test post processor run, has been updated to include the new Mapping Support global fields and their labels. The following constitutes the update:

```
10            16                                                  73
------------------------------------------------------------------------

DC            S(@MAPRCD)                                          04100001
DC            C'MAPR'                                             04100002
DC            S(@MAPFIL)                                          04100003
DC            C'MAPF'                                             04100004
DC            S(@T3270)                                           04100005
DC            C'TR70'                                             04100006
DC            S(@T3270F)                                          04100007
DC            C'TR7F'                                             04100008
```

# Record Initialization

## Pilot Tape

Only the following items are loaded from a pilot tape to each subsystem that wishes to utilize the mapping function. General pilot tapes are created by the system test compiler program (see *TPF Program Development Support Reference*).

**(GO1GO)** Main Storage Allocator Record

Contains the modifications described in Main Storage Allocator Record to accommodate the Map Sharing Table and the 3270 Translate Table.

**(AS2MT)** Map Sharing Table:

Contains the record ID MD and Zeroes

3270 Translate Table:

Contains the 256 Character Translate Table used by the Mapping Support Package as well as several peripheral programs.

The user who wishes to either increase or decrease the size of the Map Sharing Table may accomplish this by placing the desired number of *doublewords* in the RCC (Record Code Check) field of the record on the pilot tape.

## Program Initialization Of Records

The following data records are initialized by the File Create and Load Program (03-ASL), during the initial Terminal Map Record load or subsequent complete file replacements:

**AS0MP, AS1MP** Terminal Map Record

**AS4MF** Map File Index Table

## 3270 Translate Table

In order to facilitate the interrogation of lengthy data streams related to 3270 and other terminal types, this package references a 256 character table. The table is used by programs Multiple Screen Control and Input/Output Mapping to detect special control characters in messages. In addition to these programs, the Input and Output Simulator Programs reference this table. Each byte from byte 0 through byte 255 (256 byte length) contains a X'00' value, except for the following:

| Byte Relative To 0 | Position Value | Description |
|---|---|---|
| Byte 5 | X'05' | Program Tab character for 3270s and mechanical tab character for 1980 Model 24s/1977. |
| Byte 17 | X'11' | Set Buffer Address (SBA) character |
| Byte 18 | X'12' | Erase Unprotected to Address (EUA) |
| Byte 19 | X'13' | Insert Cursor (IC) character |
| Byte 21 | X'15' | Carriage Return character |
| Byte 29 | X'1D' | Start Field (SF) character |
| Byte 60 | X'3C' | Repeat to Address (RA) character |
| Byte 78 | X'4E' | End-Of-Message character |
| Byte 95 | X'5F' | End-Of-Message, Unsolicited character |
| Byte 108 | X'6C' | Line Feed character |

## Program Resident Tables and Constants

The following constitutes a list of program resident tables and constants, their use, location, and significance. In some cases, the user may desire to modify or alter these values in order to tailor the Mapping Support Package to individual system requirements.

## Program Resident Tables

| Member Name | Table | Use |
|---|---|---|
| ASLOL/ASLOM | ATTAB - Table containing 3270 attributes. | Used to convert user supplied attribute information to valid 3270 attributes. |
| | TABLE - Table containing valid 3270 buffer addresses. | Used to convert position supplied by the user to a valid 3270 buffer address when a 3270 device is specified. |
| ASC4 | Table containing 3270 buffer address characters. | To convert locations expressed as relative positions to their equivalent two character 3270 addresses. |
| ASFJ, ASFP | Table containing buffer addresses for system lines for associated device types. | To insert a terminal devices system line addresses, characters/line and lines/screen when creating a *pseudo* RCB. |
| ASL2 | Table containing terminal type indicators and associated record type and Map Index Table (AS4MF) ordinal number. | To retrieve relevant Map Index (AS4MF) for update by Map File Create and Load. To pass to Mapping and Map File Create and Load device record type.<br><br>Example of item in ASL2 Device Table:<br>`line 1 DC XL2'10C1'`<br>`line 2 DC X'00'`<br>`line 3 DC AL1(#TER01)`<br><br>Where:<br>`line 1 byte 0 • Terminal type indication as found in CI0CO (CI0TTP)`<br>`byte 1 • Alpha designation per device as assigned by offline map file create (ASLOL)`<br>`line 2 byte 0 • Ordinal number assigned by user for this device types' Map Index record (AS4MF)`<br>`line 3 byte 0 • Resolved record type assigned in SYSEQ for a particular device type.` |
| ASL6 | Table containing mapping error messages. | User gives ASL6 error message number. Table is searched for match, and message is sent to user specified terminal or application. |
| ASL5 | Table containing the record types of all devices supported by mapping. | At map file create time, the files are initialized with the appropriate ID and the remainder of the header is zeroed out. |

## Program Constants and Literals

| Member Name | Constants/Literal | Use |
|---|---|---|
| ASF2 | =X'D6D4' | Record ID for output AMSG. |
| ASF3 | Same | Same |

| Member Name | Constants/Literal | Use |
|---|---|---|
| ASF4 | =X'D4C900' | Record ID and Code Check for formatted user DSECT. |
| | =X'D4E200' | Record ID and Code Check for MSP (Terminal Map Rcd). |
| | =X'D4C200' | Record ID and Code Check for AS5MB (Page/Scroll Control Record). |
| | =X'D6D400' | Record ID and code check for AMSG output block. |
| | =C‘SDLC’ | Designation of a *pseudo* RCB on level 3. |
| ASF8 | =X'D4C900' | Record ID and Code Check for input AMSG block. |
| | =X'D6D400' | Record ID and RCC for output AMSG block |
| | =C‘SDLC’ | Designation for a *pseudo* RCB on level 3. |
| ASFI & ASFO | =H‘519’<br>=X'D4C100' | AS4MF item count.<br>Record ID and Code Check for AS4MF. |
| | =X'D4E200' | Record ID and Code Check for MSP. |
| ASF5 | =X'D4C900' | Record ID and RCC for AMSG overflow record retrieval. |
| ASCA | =X'D4C200' | Record ID and RCC for AS5MB (Page/Scroll Control Rcd). |
| | =X'D4C900' | Record ID and RCC for input AMSG block. |
| ASLOL/ASLOM | (RID) C‘MS’ | Record ID of MSP record. |
| ASLOL/ASLOM | (CID) C‘MR’ | ID of Control Record on MRT tape. |
| ASLOL/ASLOM | (COMMAND) X'F8' | Write command default for 3270 devices. |
| ASLOL/ASLOM | (WCCBYTE) B‘00000010’ | Default WCC for 3270 devices. |
| ASLOL/ASLOM | (ATTRB) B‘01100000’ | Default attribute for 3270 devices. |
| ASLOL/ASLOM | (ATTRB2) B‘00000010’ | Part 2 of attribute for output maps. |
| ASL1 | =X'D4E200' | ID, Record Code Check for Terminal Map Record. |
| ASL2 | =‘D4C100’ | Record ID, Record Code Check for Map Index Table. |
| ASL3 | =X'D4E200' | Record ID and Record Code Check for Terminal Map Record. |
| | =C‘MR’ | Record ID of Tape File. |
| ASL5 | =X'D4C100' | Record ID and Record Code Check for Map Index Table. |
| | =X'D4E200' | Record ID and Record Code Check for Terminal Map Record. |
| ASC1 | =X'D6D400' | Record ID and Record Code |
| | =X'D4C200' | Record ID and Record Code Check for Page/Scroll Control |
| | END DC H‘337’ | Data area length for output AMSG. |
| | C‘SDLC’ | Designation for *pseudo* RCB on level 3 |
| | =X'FFFFFFFF' | Designation for ECB save area on level 4. |

| Member Name | Constants/Literal | Use |
|---|---|---|
| ASC2 | =X'D4C900' | Record ID and Record Code Check for input AMSG. |
| | =X'D4C200' | Record ID and Record Code Check for Multiple Screen Control Record. |
| ASC3 | =X'D6D400' | Record ID and Record Code Check for output AMSG. |
| | END DC H‘337’ | Data area length for output AMSG. |
| | DC XL7‘1140403C000040’ | 3270 repeat to address command. |
| | =F‘63’ | Constant needed to convert 3270 buffer address to relative positions. |
| ASC4 | =X'D4C900' | Record ID and Record Code Check for input AMSG. |
| | =X'D6D400' | Record ID and Record Code Check for output AMSG. |
| | END DC H‘337’ | Data area length for output AMSG |
| ASL4 | =X'D6D400' | Record ID and Record Code Check for output AMSG. |
| | =X'5CF1' | Address for 3270 Model 2 terminal. |
| | =X'C6F9' | Address for 3270 Model 1 terminal. |
| | =X'F1C611' | WCC, CMD and SBA for 3270 terminal. |
| | =X'0000' | Constant to test for presence of address in CI0RPL. |
| ASC5 | =X'D6D400' | Record ID and Record Code check for output AMSG. |
| | DC H‘337’ | Data area length for output AMSG. |
| | DC XL7‘1140403C000000’ | 3270 repeat to address command. |
| | =F‘63’ | Constant needed to convert 3270 buffer address to relative position. |
| ASC6 | =X'D6D400' | Record ID and Record Code Check for output AMSG. |
| | =F‘63’ | Constant needed to convert 3270 buffer address to row, column format. |
| | DC XL7‘1100001D406E4E’ | SBA, address, SF, >, ATT and EOM needed for 3270 terminal (system line). |
| ASC8 | DC XL7‘1140403C000000’ | 3270 Repeat to address order. |
| | DC XL10‘C6F80011C6F83C4DF000’ | Erase order for upper part of 1920 split screen. |
| | DC XL11‘4F403CD5F80011D5F83C5C’ | Erase order of upper part of 1920 split screen. |

## User Modifications To Package Parameters

The terminal types supported may exceed current user requirements. Thus, each individual installation may need alteration of the tables and file allocations to delete terminal type references. The deletion of references to supported terminal types may occur in either of two ways: (1) the user may delete *all* references to the supported terminal by updating the FACE Table, the System Equate Macro, the FACE Driver and Offline Interface Program, etc., or (2) allow the references to remain unused, thus allowing for possible future terminal type expansion. The following description is a guide to the process involved with the former alternative.

1. Delete the RA#TERnn equate statement that corresponds to the AS supported terminal type from the System Equate Macro (SYSEQ).
2. Delete the RAMFIL statements in the SIP Stage I deck that correspond to the #TERnn equate in SYSEQ.
3. Update the Recoup Package descriptor to remove the reference to the file deleted from the FACE Table. See *TPF Database Reference* for more information.
4. Delete the reference and alpha identifier from various program resident tables and macros provided with the TPF Mapping Support Package.
   a. ASLOL - This program contains code and constants for the purpose of validating user specified device types for map record creation.
   b. ASL2 - This program contains a table of terminal related items; each item in this table contains the #TERnn equate, the MSI record ordinal, the unique alpha identifier, and the one character hexadecimal RCB notation (Reference field CI0TTP) which correspond to each terminal supported by mapping. There is one such item for each terminal type.
   c. ASL5 - This program contains a table of terminal equates (#TERnn), one for each terminal supported by mapping.
   d. DPANL - This macro contains macro language statements which are used for the purpose of validating each terminal type supported by this package.

   New devices may be added by reversing the above process. However, the user should exercise care in this process and determine if the device characteristics (field oriented, for example) and control characters are compatible with the formatting capability of this package. The formatting and mapping function of this package is primarily directed towards the 3270 Information Display System and its related terminal (CRT and Printer) features.

# Mapping Support User and Operating Procedures

## Introduction

This section describes the process and procedures necessary to (a) create, load, utilize, and maintain the online Terminal Map Record files, and (b) generate symbolic reference modules (DSECTS) in user assembled programs. In addition, the terminal inputs required to utilize paging and scrolling, and their resultant responses, are detailed. The information set forth includes input preparation, system error and message interpretation, general process flow, input messages and associated responses, and the job control language (JCL) necessary to invoke the offline phases of processing.

## Physical/Symbolic Map Create Process

The process of creating and defining map formats results in the generation of *symbolic* reference modules (DSECTs) in user assembled programs and the initialization and maintenance of the *physical* online map records used by the Input/Output Mapping Program. The following steps outline the entire process involved in order to obtain the final results.

1. The user decides on the exact contents and format of a printout or CRT display. This may be defined by a desired report design or function design. The user describes, on graph paper or on an IBM *3270 Information Display System Panel Layout Sheet*, the fields that are to appear in the panel, the attributes and characteristics of each field, and the text that is to appear on each panel.

2. After laying out the panel, the information is transcribed to coding sheets. The various panel contents and field characteristics are written as components of macro keyword parameters. Each panel and group of fields comprising that panel is represented by a series of parameterized DPANL and DDATA map macro source statements. In addition, each panel is assigned, by the user, a unique symbolic name and sequence number.
3. The information previously recorded on coding sheets is punched into cards.
4. Each group of map source statements, which corresponds to a panel, is composed of a series of one or more DDATA statements. There is one DPANL statement preceding the first DDATA card and one following the last DDATA card; this sequence constitutes a map. Each uniquely named and numbered map is added, by the user, to a Map Source Statement Library using a standard MVS Source Library Edit utility. The Map Source Statement Library, which is allocated by the user as a partitioned data set, contains all the map source statements (DPANL and DDATA) and should be the only storage medium where these statements reside.
5. Utilizing the symbolic name assigned to each field comprising a map, the user codes his ECB controlled source program to perform a desired function. Within the source code, the user will write one or more *COPY mapname* statements. Each COPY statement will cause the assembler to insert the requested map source statements from the library into the source program immediately after the COPY statement and to generate an associated DSECT. The *mapname* operand is the same as the member name in the source library.
6. The source program, including COPY statements, is input to the MVS Assembler. These copy members must be on the MVS SYSLIB data set. The result of the assembly is an object module and program listing. The latter will contain an expansion of the DSECT, informational comments, and error messages, if applicable. Incorrectly coded or missing DPANL and DDATA parameters result either in default assumptions being taken or assembly errors being initiated. The user should use the assembly listing as a guide to correct errors and incorrectly coded map specifications. The corrected map source statements are then replaced in the library and the user source program is reassembled. The result of the successful assembly is a DSECT consisting of symbolically addressable fields.
7. The Map Source Statement Library is used as input to the offline Map File Create and Load Program (03-ASL). When a file create or replacement has been requested, this segment (ASLOL) sequentially reads the source statements in the library, beginning with the first DPANL statement, then proceeding to the DDATA field definitions, and finally concluding with the ending DPANL statement, for each map (ASLOL in this topic refers to both offline segments ASLOL and ASLOM). That is, for the addition, deletion, or replacement of specific MSP records in a file, segment ASLOL obtains the uniquely assigned map name or map sequence number from the input cards and locates the associated map in the Map Source Statement Library. ASLOL inspects each statement for valid order, format, and data; errors are written to the printer. If no error is encountered during the edit of a map, ASLOL will construct the corresponding Terminal Map Record and write the formatted record to the Map Macro Tape File (MRT). During this phase of the Map File Create processing, the user may invoke a function which provides two listings of the map names and sequence numbers currently residing in the Map Source Statement Library. The first listing consists of the symbolic map names, in ascending alphabetic order, and their associated sequence numbers. The second listing contains the symbolic map names and sequence numbers and is arranged in ascending order, by map sequence number.

8. Upon completion of MSP file create or replacement, the user invokes the SORT Utility Program to arrange the Terminal Map Records by ascending map sequence number within each device type. The sorted records are written to the Map Macro Tape File (AS0MP), which will be used as input to the online portion of the Map File Create and Load Program.
9. A Z-type command invokes the online segment of the Map File Create and Load Program which sequentially reads the MSP Records from the MRT File and writes each record to its respective file location. During this phase of processing, the mapping and formatting function is inhibited. Any errors encountered during the load phase are recorded on the PRC high speed RO device. At the conclusion of the load phase, mapping and formatting are again allowed, and a map load completed response is sent to the terminal which initiated the command. The command is ZAMAP. With the Multiple Data Base Facility, to load into a particular subsystem, prefix the command with the subsystem ID.

## Macro Description

The Mapping Support macro instructions are used in application development to declare a data structure used in processing terminal displays. DPANL and DDATA macro instructions (called map statements) written by an application programmer must be placed onto a Map Source Statement Library with an MVS utility program. Map statements are then included in ECB controlled source code through the use of the assembly language COPY instruction. The copied map statements have the effect of declaring symbolic names used in a display data structure, since the DPANL and DDATA instructions expand into a DSECT included in the source code. The DSECT is adjusted and a register is assigned as the result of parameters included in the map statement coding.

The effect of using map statements is analogous to the use of a data macro instruction. However, map statements do *not* appear in the source code of ECB controlled programs. This means that source code does not contain the register assignment of the DSECT generated as the result of copied map statements.

A map statement on the Map Source Library is used in two different ways:

- A map statement is copied into an ECB controlled program source code as outlined above.
- A map statement is used by a portion of the Mapping Support Package to create a physical Terminal Map Record in the fixed file area. These records are used by the system supplied ECB controlled programs to superimpose or delete control characters in data streams processed by application programs.

### Terminal Panel Display Macro (DPANL)

This macro, which consists of parameterized keywords coded by the user, provides the capability for the user to specify, for each format or panel, a unique map name and sequence number, a related terminal type, 3270 command and control information, and DSECT size. There are two of these statements for each user written map.

### Terminal Data Field Display Macro (DDATA)

This macro statement, which is coded by the user, defines the symbolic name, length, characteristics, and position of each field in a user designated report or display. The inclusion of the field name provides the user symbolic addressability to a specific area of a DSECT. There is a single DDATA statement for each field appearing in a user desired report or display.

## Record Description

### Terminal Map Record

This record is constructed offline by ASLOL. The data that is contained in each record is obtained from the keyword parameters of the DPANL and DDATA macro source statements previously written by the user. After the MSP Records have been constructed and written to the MRT File, they are loaded by the online ASL program for use by the Input/Output Mapping Program to format data streams. There is *one* physical Terminal Map Record created for each terminal type specified on the DPANL macro statement of a specifically named and numbered map.

## Creating Map Source Statements

### Terminal Panel Display Macro (DPANL)

There are two (2) DPANL statements required for each user defined map. The first precedes and the latter follows the DDATA field definitions.

The following examples of DPANL statements and their associated MVS Assembler generated parameters are provided as representative samples of this macro. Note that a map definition consists of DPANL *and* DDATA statements; only examples of the former are shown at this point.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

MAP1      DPANL   DSECT=START,                                        X
                  MODE=IN,                                           X
                  SEQNO=75,                                          X
                  MAXFLDS=10
+MAP1I    DSECT
+MAP1IHDR DS      0CL16         MESSAGE BLOCK HEADER
+MAP1IBID DS      CL2           RECORD ID
+MAP1ICHK DS      CL1           RECORD CHECK
+MAP1ICTL DS      CL1           RECORD CONTROL FIELD
+MAP1IPGM DS      CL4           MODIFYING PROGRAM FIELD
+MAP1IFCH DS      CL4           FORWARD CHAIN ADDRESS
+MAP1ILIT DS      CL4           LINE IA TA & CPUID
+MAP1IMF  DS      CL2           INPUT FIELD BIT INDICATORS
          ---
          ---
          ---
          ---


          DPANL   DSECT=END,REG=R1
+         USING   MAP1I,R1
+ IS      CSECT
+#MAP1I   EQU     75            MAP SEQUENCE NUMBER
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

MAP2      DPANL   DSECT=START,                                        X
                  MODE=OUT,                                          X
                  SEQNO=128,                                         X
                  DEV=(3277-1,3286-1),                               X
                  OMSGBLK=128,                                       X
                  CMD=W
+MAP2O    DSECT
+MAP2OHDR DS      0CL16         MESSAGE BLOCK HEADER
+MAP2OBID DS      CL2           RECORD ID
+MAP2OCHK DS      CL1           RECORD CHECK
+MAP2OCTL DS      CL1           RECORD CONTROL FIELD
+MAP2OPGM DS      CL4           MODIFYING PROGRAM FIELD
+MAP2OFCH DS      CL4           FORWARD CHAIN ADDRESS
+MAP2OLIT DS      CL4           LINE IA TA & CPUID
```

```
          ---
          ---
          ---
          DPANL   DSECT=END,REG=R4
+         USING   MAP2O,R4
+ IS      CSECT
+#MAP2O   EQU     128           MAP SEQUENCE NUMBER
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

MAP3      DPANL   DSECT=START,MODE=BOTH,SEQNO=(3,4),MAXFLDS=17
          DDATA
          DPANL   DSECT=END,REG=R4

+MAP3I    DSECT                 MSG BLOCK DSECT
+MAP3IHDR DS      0CL16         MSG BLOCK HEADER
+MAP3IBID DS      CL2           RECORD ID
+MAP3ICHK DS      CL1           RCD CHECK
+MAP3ICTL DS      CL1           RCD CONTROL FIELD
+MAP3IPGM DS      CL4           MODIFY PGM FIELD
+MAP3IFCH DS      CL4           FWD CHAIN FIELD
+MAP3ILIT DS      CL4           LINE IA TA & CPU ID
+MAP3IMF  DS      CL3           MODIFY FIELDS INDICATORS
          DS      GENERATED BY DDATA - REFERENCE DDATA MACROS
+         USING   MAP3I,R4
+ IS +     CSECT
+#MAP3I    EQU     3             MAP SEQUENCE NUMBER
          ---
          ---
          ---
          ---
          ---

+MAP3O    DSECT                 MSG BLOCK DSECT
+MAP3OHDR DS      0CL16         MSG BLOCK HEADER
+MAP3OBID DS      CL2           RECORD ID
+MAP3OCHK DS      CL1           RCD CHECK
+MAP3OCTL DS      CL1           RCD CONTROL FIELD
+MAP3OPGM DS      CL4           MODIFY PGM FIELD
+MAP3OFCH DS      CL4           FWD CHAIN FIELD
+MAP3OLIT DS      CL4           LINE IA TA & CPU ID
+MAP3ORES DS      CL3           NOT USED
          DS      GENERATED BY DDATA - REFERENCE DDATA MACRO
          USING   MAP3O,R4
+ IS      CSECT
+#MAP3O   EQU     4             MAP SEQUENCE NUMBER
```

## Terminal Data Field Display Macro (DDATA)

There is a variable number of DDATA statements for each user defined map. Generally, there is one (1) DDATA statement for each field the user desires to define for input or output data streams. For a detailed description of the various keywords and parameters to be declared by using the DDATA macro, see *TPF General Macros*.

The following examples of DDATA statement and their associated MVS assembler generated parameters are provided as representative examples of this macro.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

ABCD     DDATA    POS=(1,1),LENGTH=10,REPEAT=4 (input map)
ABCD0L   DS       H          DATA FIELD INPUT LENGTH
ABCD0I   DS       CL10       INPUT DATA
ABCD1L   DS       H
ABCD1I   DS       CL10
```

```
ABCD2L   DS       H
ABCD2I   DS       CL10
ABCD3L   DS       H
ABCD3I   DS       CL10
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

XYZ      DDATA    POS=(5,10),LENGTH=3,REPEAT=2 (output map)
         DS       0H
XZY0A    DS       CL1         ATTRIBUTE BYTE
         DS       CL1         RESERVED
XYZ0O    DS       CL3         OUTPUT DATA
         DS       0H
XYZ1A    DS       CL1         ATTRIBUTE BYTE
         DS       CL1         RESERVED
XYZ1O    DS       CL3         OUTPUT DATA
```

In this case, the first field will be positioned at row 5, column 10 of a display/print; the second field (XYZ1A + XYZ1O) will be at row 6, column 10.

## MNOTE Messages Produced By DPANL and DDATA Macros

The MNOTE messages produced by the DPANL and DDATA macros are self-explanatory. The severity code associated with each MNOTE message indicates whether corrective action is necessary or not. A severity code of 2 indicates that a default value is assigned to an omitted keyword parameter. A severity code of 4 indicates a condition which must be corrected. A severity code of 0 is an information message.

## Offline Map File Create and Modify

The following section details the JCL requirements, hardware, card input, and messages associated with the offline portion of the Mapping Support. This phase of the package provides the user with the means of creating and replacing entire Terminal Map Record (MSP) files, and of adding, deleting, and replacing individual MSP records residing in a file.

### Hardware Requirements

System hardware requirements for the execution of the offline version of the Map File Load/Create program are as follows.

### For the Create Run

| Device | #REQ | Device Name | Purpose |
|---|---|---|---|
| CARD READER | 1 | OS/READER SYSIN | Read JCL to initiate program ASLOL. Sort program/print program. |
| PRINTER | 1 | OS/PRINTER SYSPRINT/SYSUT2 | Print JCL error messages/ utility messages. |
| | | PRINT | Name given printer device by program ASLOL to log program error messages. |
| DISKS | 1 (min) | MAPD/MAPS | Names given users Map Source Library. |
| | | SORTWK01,02,03 | Disk workspace for Sort program. |

| Device | #REQ | Device Name | Purpose |
|---|---|---|---|
| TAPE | 1 | MAPTAPE | Name given MRT tape to be created by program ASLOL. |
| | | SORTIN/SORTOUT | Same MRT tape created by ASLOL but used as input/output to Sort program. |
| | | SYSUT1 | Same MRT tape created by ASLOL and sorted by Sort. Now used as input to Print/ Punch utility. |

### For The Modify Run

Hardware requirements for the modify phase are the same as the Create run, with the following exception. The card reader will be used by the ASLOL program as an input device with the logical name `Card` needed in the JCL run (see "JCL Examples").

## JCL Examples

### Create Run

```
//CREATE JOB
//JOBLIB DD DSN=P8740SM1.VSM1.PTF0037.SORTLIB,DISP=SHR
//MAPINIT EXEC PGM=IEHINITT       (INITIALIZE MRT TAPE)
//SYSPRINT DD SYSOUT=A
//LABEL DD UNIT=(TAPE,,DEFER)
//SYSIN DD *
LABEL    INITT   SER=111111
/*
//MAPGEN EXEC PGM=ASLOL,PARM='CREATE,PRINT',TIME=5  (MRT CREATE)
//STEPLIB DD DSN=USER LINK LIBRARY,DISP=(SHR,PASS)  (ASSUMED CATALOGED)
//PRINT DD SYSOUT=A,DCB=(RECFM=FB,BLKSIZE=3630,LRECL=121)
//MAPD   DD DSN=USER MAP LIBRARY,DISP=SHR  (ASSUMED CATALOGED)
//MAPS    DD DSN=USER MAP LIBRARY,DISP=SHR  (ASSUMED CATALOGED)
//MAPTAPE DD DCB=(BLKSIZE=1055,LRECL=1055,RECFM=FB),DSN=MAPRCDS,
//        UNIT=(TAPE,,DEFER),VOL=SER=111111,DISP=(OLD,PASS),
//        LABEL=(,SL)
//SYSUDUMP DD SYSOUT=A
//PRINT2 DD SYSOUT=A,DCB=(RECFM=FBA,BLKSIZE=3630,LRECL=121)
//SYSOUT DD SYSOUT=A
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
//STORE DD DSN=&&PRTDATA,DCB=BLKSIZE=18,LRECL=18,RECFM=FB),
//      UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG),DISP=(NEW,DELETE)
//SORTIN DD DSN=&&PRTDATA,DISP=(,PASS),VOL=REF=*.STORE
//SORTOUT DD DSN=&&PRTDATA,DISP=(OLD,DELETE),VOL=REF=*.STORE
//SORTWK01  DD  UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG)
//SORTWK02  DD  UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG)
//SORTWK03  DD  UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG)
//CARD   DD *
/*
//SORT   EXEC SORTD  (MRT SORT)
//SORT.SORTIN DD DSN=MAPRCDS,UNIT=(TAPE,,DEFER),VOL=SER=111111,
//     DCB=(LRECL=1055,BLKSIZE=1055,RECFM=FB),
//     LABEL=(,SL),DISP=OLD
//SORT.SORTOUT DD UNIT=AFF=SORTIN,DSN=MAPRCDS,VOL=SER=111111,
//     DCB=(LRECL=1055,BLKSIZE=1055,RECFM=FB),LABEL=(,SL),
//     DISP=(NEW,PASS)
//SORT.SORTWK01 DD UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG),
//       VOL=REF=*.MAPGEN.SORTWK01
//SORT.SORTWK02 DD UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG),
         VOL=REF=*.MAPGEN.SORTWK02
//SORT.SORTWK03 DD UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG),
         VOL=REF=*.MAPGEN.SORTWK03
//SORT.SYSIN DD *
```

```
SORT FIELDS=(23,1,CH,A,13,2,BI,A,5,4,CH,A,9,2,BI,A),SIZE=E120
END
/*
//PRINT EXEC PGM=IEBPTPCH  (PRINT FINAL SORTED MRT)
//SYSUT1 DD DCB=(BLKSIZE=1055,LRECL=1055,RECFM=FB),DSN=MAPRCDS,
//       UNIT=(TAPE,,DEFER),VOL=SER=111111,DISP=(OLD,KEEP),
//       LABEL=(,SL)
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD SYSOUT=A
//SYSIN DD  *
    PRINT   TOTCONV=XE
/*
//
```

## Modify Run

```
//MODIFY JOB
//JOBLIB DD DSN=P8740SM1.V3M1.PTF0037.SORTLIB,DISP=SHR
//MAPINIT EXEC PGM=IEHINITT       (INITIALIZE MRT MOD TAPE)
//SYSPRINT DD SYSOUT=A
//LABEL DD UNIT=(TAPE,,DEFER)
//SYSIN DD *
LABEL    INITT  SER=111111
/*
//MAPGEN EXEC PGM=ASLOL,PARM='MODIFY,PRINT',TIME=8
//STEPLIB DD DSN=USER LINK LIBRARY,DISP=(SHR,PASS)  (ASSUMED CATALOGED)
//PRINT DD SYSOUT=A,DCB=(RECFM=FB,BLKSIZE=3630,LRECL=121)
//MAPD    DD DSN=USER MAP LIBRARY,DISP=SHR  (ASSUMED CATALOGED)
//MAPS     DD DSN=USER MAP LIBRARY,DISP=SHR  (ASSUMED CATALOGED)
//MAPTAPE DD DCB=BLKSIZE=1055,LRECL=1055,RECFM=FB),DSN=MAPRCDS,
//        UNIT=(TAPE,,DEFER),VOL=SER=111111,DISP=(OLD,PASS),
//        LABEL=(,SL)
//SYSUDUMP DD SYSOUT=A
//PRINT2 DD SYSOUT=A,DCB=(RECFM=FBA,BLKSIZE=3630,LRECL=121)
//SYSOUT DD SYSOUT=A
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
//STORE  DD DSN=&&PRTDATA,DCB=(BLKSIZE=18,LRECL=18,RECFM=FB),
//       UNIT=SYSDA,SPACE=TRK,(20),,CONTIG),DISP=(NEW,DELETE)
//SORTIN DD DSN=&&PRTDATA,DISP=SHR,VOL=REF=*.STORE
//SORTOUT DD DSN=&&PRTDATA,DISP=(OLD,DELETE),VOL=REF=*.STORE
//SORTWK01   DD  UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG)
//SORTWK02   DD  UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG)
//SORTWK03   DD  UNIT=SYSDA,SPACE=(TRK,(20),,CONTIG)
//CARD  DD  *
D       299  B   DELETE MAP WITH SEQ NO. OF 299 FOR DEV TYPE B
D       2300 BF  DELETE MAP WITH SEQ NO. OF 2300 FOR DEV TYPE B & F
D       3000 BF  DELETE MAP WITH SEQ NO. OF 3000 FOR DEV TYPE B & F
D       2800 ABC DELETE MAP WITH SEQ NO. OF 2800 FOR DEV TYPE A, B & C
A       AAAA     ADD MAP NAMED AAAA
A       MAP      ADD MAP NAMED MAP
A       LARG     ADD MAP NAMED LARG
A       RCDS     ADD MAP NAMED RCDS
R       TKT1     REPLACE MAP NAMED TKT1
R       PAGE     REPLACE MAP NAMED PAGE
R       LARG     REPLACE MAP NAMED LARG
R       ZZZZ     REPLACE MAP NAMED ZZZZ
/*
//PRINT EXEC  PGM=IEBPTPCH  (PRINT MRT MOD TAPE)
//SYSUT1  DD  DCB=(BLKSIZE=1055,RECFM=FB),DSN=MAPRCDS,
//        UNIT=(TAPE,,DEFER),VOL=SER=111111,DISP=(NEW,KEEP),
//        LABEL=(,SL)
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD SYSOUT=A
//SYSIN DD  *
    PRINT   TOTCONV=XE
/*
//
```

# Input Requirements For Program (ASLOL)

## Create Run

For the Map File Create program, the input requirements for a Create run are as follows:

1. The user-coded DPANL and DDATA statements installed on the user's Map Source Library.
2. The JCL as shown in the JCL example.

## Modify Run

The input requirements for the modify run of the Map File Load/Create run are as follows:

1. The user-coded DPANL and DDATA statements installed on the user's Map Source Library.
2. The JCL as shown in the JCL example.
3. The Modify Control Cards. Reference Modify Control Card Specifications and examples.

## Modify Control Card Specifications

## Delete a Map

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

D        Map    Device Type Code
         Seq. #
```

**Note:** More than 1 Device Type Code can appear on 1 card starting in column 16 and ending before column 72.

## Modify an Existing Map

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

R        Map Name
```

## Add a New Map

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

A        Map Name
```

## Modify Run Examples

Delete a map for Device Type A (3277-1) with sequence number 123.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

D        123   A
```

Delete a map for Device Types A and B (3277-1 and 3277-2) with sequence number 156.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

D        156   A B
```

Modify the existing map named Map1 with the version now on the Map Source Library.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

R        MAP1
```

Add to the existing map file the map named MAP2 now in the Map Source Library.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

A        MAP2
```

## Error and Communication Messages

All error and communication messages issued by the offline generic mapping support program are full length descriptive messages and should provide the user with information about what action to take in order to correct the problem. Should the user have difficulty understanding the messages and what action, if any, to take, see *Messages (System Error and Offline)* and *Messages (Online)*.

## Input/Output Mapping User Requirements

To format (map) input and output data streams, the user written application program segment must request activation of the Input/Output Mapping Program. This is accomplished by means of an ENTER type macro to members ASFO/ASFI for RCB associated devices or to ASFP/ASFJ for devices not associated with an RCB. In addition, the user passes information in a variable length work area and the unformatted data stream to the mapping function. A more detailed explanation of these points follows.

### Input Mapping

This phase of processing reformats native mode data streams emanating from the IBM 3270 Information Display terminal. The native mode data, includes device dependent control characters (buffer address and buffer order characters) interspersed among the entered text. Input mapping deletes from the data stream the device dependent control characters inserted by the 3270 control unit and places the agent/operator entered text in specific areas of blocks of main storage; these areas coincide with a previously defined symbolic reference module (DSECT). In addition, a series of bit indicators, one for each possible input field, is used by the input mapping function to summarize, to the user application, the contents of each respective input field. When text is entered for a field, the corresponding bit is set on.

These bit indicators are stored in the field suffixed by IMF in the user DSECT.

The reformatted block is then returned to the user via the BACKC macro or passed to a user determined destination via the ROUTC macro.

The user written application program segment entry requirements to input mapping are summarized as follows:

- The prime input message block must be on data level 0; overflow records, if any, are on file.
- The mapping interface work area has been correctly initialized with the map sequence number and RCPL (Routing Control Parameter List), if the latter is required.
- The appropriate ENTER type macro must be executed. When the user application segment is to regain control after the input data has been formatted, an ENTRC must be coded. When the user application does not want to be returned to after the input data stream has been mapped, an ENTNC must be coded.

It is the user application's responsibility to determine if the device has an associated RCB. If the device has an associated RCB, the user application codes an ENTRC ASFI to regain control or codes ENTNC ASFI if not. If the device *does not* have an

associated RCB, the user application codes an ENTRC ASFJ to regain control or codes an ENTNC ASFJ if no return is desired.

- Register 1 must contain the address of the start of the mapping interface work area, which is required to begin on a halfword boundary.

## When an ENTRC Has Been Issued

| | |
|---|---|
| Bytes 0–1 | Contain the map sequence number in binary representation. |
| Byte 2 | For return, must be set to a X'00'. On return from ASF, contains X'00' for nonerror condition or binary value for error condition. |
| Bytes 3–5 | Reserved for use by Mapping Support Package.<br>**Note:** In this case, the input data is reformatted and returned to the user. No ROUTC is issued by the AS package. |

## When an ENTNC Has Been Issued

| | |
|---|---|
| Bytes 0–1 | Contain the map sequence number in binary representation. |
| Byte 2 | For no return, must be set to a X'80'. |
| Bytes 3–5 | Reserved for use by the Mapping Support Package. |
| Bytes 6–17 | This represents the Routing Control Parameter List (RC0PL) that is used to route messages. |
| Bytes 6–9 | Contain the message destination, either a terminal address and processor ID or a symbolic application name. |
| Bytes 10–13 | Contain the symbolic application name origin of this message. |
| Bytes 14–17 | Contain control information - Reference RC0PL.<br>**Note:** When byte 2 =X'80', the AS package issues the ROUTC. |

An example of code necessary to invoke the input formatting function may consist of the following:

```
BEGIN
  --
  --
  --
  --
 LA    R1,#MAP1I      MAP SEQUENCE NUMBER
 STH   R1,EBW000      STORE IN WORK AREA
 MVI   EBW002,X'00'   INDICATE RETURN
 LA    R1,EBW000      ADDR OF WORK AREA
 ENTRC ASFI
 CLI   2(R1),X'00'    ERROR RETURN
 BNE   ERROR          YES
 L     R2,CE1CR0      USER DSECT
 TM    MAP1IMF,X'40'  SECOND INPUT FIELD CONTAIN DATA
 BO    TAG1           YES
 TM    MAP1IMF,X'01'  EIGHTH INPUT FIELD CONTAIN DATA
 BZ    TAG2           NO
 --
 --
 --
 --
 --
 FINIS
```

## Additional Considerations

- The mapping interface work area must *not* reside in the input message block on level 0.

- Input mapping (ASFI) will obtain pertinent terminal type information by utilizing the LNIATA or RID contained in EBROUT and the CPU ID contained in CE1CPD; both fields, which reside in the Entry Control Block (ECB), are used to access the associated CI0CO from which the appropriate terminal characteristics are obtained.
- When input mapping (ASFI/ASFJ) is invoked and the input message block on level 0 contains an agent/operator initiated page or scroll request, the requested page is formatted by the ASC program segments. The corresponding screen is displayed to the agent/operator by *AS* via the ROUTC *only if* the user application has indicated such in the mapping interface work area (Byte 2-X'80'). If return is requested by the user application, it is the user's responsibility for displaying the corresponding screen. The presence of a page/scroll request in the input message does *not* mean the formatted page is transmitted by the AS package. The user must request the ROUTC to be issued by AS.
- It should be reemphasized that the user application is responsible for determining which segment of mapping to activate for input processing:

  **ASFI** input mapping for terminals associated with a RCB

  **ASFJ** input mapping for terminals *not* associated with an RCB
- Upon return to the user, each item in the DSECT will consist of the *actual* length of the entered text and the text. When data has not been entered, the halfword length field contains a value of zero (X'0000').
- On return from input mapping the user must check data level 0. If there is a block on level 0 it will contain either the first or only block of mapped text. The lack of a block on level 0 will indicate that the input was a page request and nothing has been mapped yet (the output has been paged). If the maps contain no modified fields (display only) the data level will be free upon return to the application.

## Output Mapping

This phase formats for display on 3270 CRTs and for printing on hard copy terminals output messages which have been constructed by the user application program. For 3270 type terminals, whether CRT or printer, output mapping reformats the user created data stream, which consists solely of text, and constructs an output message containing device dependent control characters. These control characters consist of 3270 commands, orders, and attributes. The user created text is resident in a block(s) of storage which coincides with a program member assembled symbolic reference module (DSECT).

For non-3270 type devices supported by output mapping, data fields are placed in their defined positions by using blanks (X'40') as fill characters or, for those devices equipped with a mechanical tab feature (for example, 1980 Model 24), tab control characters (X'05'). When fields are placed on new lines, carriage control characters (X'15') are inserted into the output data stream. The positioned data is obtained from the user program generated DSECT.

After the output data has been mapped, the blocks are either returned to the user via the execution of a BACKC macro, or the data is routed to a terminal/application via the ROUTC macro.

The user written application program segment entry requirements to output mapping are summarized as follows:

1. The prime user block (DSECT) must be on data level 0. When overflow records exist, these are on file.

2. The mapping interface work area has been correctly initialized with the map sequence number and RCPL (Routing Control Parameter List); the latter when control is *not* returned to the requesting program. This work area has already been discussed in the previous section.
3. The appropriate ENTER type macro must be executed. When the user application segment is to regain control after the output data has been reformatted, an ENTRC must be coded. When the user application does not want control returned after the output data has been mapped, an ENTNC must be coded. It is the user application's responsibility to determine if the device has an associated RCB. If the device has an associated RCB, the user application codes an ENTRC ASFO to regain control or codes an ENTNC ASFO if not. If the device does not have an associated RCB, the user codes an ENTRC ASFP to regain control or codes an ENTNC ASFP if mapping is to route the formatted block. In this case, the interface work area must contain a user created RCPL. The Output Mapping function will issue a ROUTC.
4. The mapping interface work area must *not* reside in the DSECT block on level 0.
5. Register 1 must contain the address of the start of the mapping interface work area, which begins on a halfword boundary.
6. Bytes 12–15 of the prime user block (DSECT) on level 0 must contain the symbolic LNIATA or RID and the CPU ID of the terminal against which the output data is to be mapped. This field is symbolically addressable by the tag *mapname* OLIT (e.g. MAP2OLIT).

   **Examples**:

   For LNIATA: Bytes 12–15 520302C1

   For RID: Bytes 12–15 001006C1
7. The user is responsible for obtaining the block(s) of storage which will contain his data. This is the block that is passed to ASFO/ASFP on level 0. The user is also responsible for insuring that the portions of his work block which do not contain data are initialized with binary zeros (X'00'). This is necessary because output mapping interrogates each field of the user DSECT for the existence of data; the presence of binary zeroes implies *no* data for that specific field. If fields which do not contain data are not set to binary zeroes, the results are unpredictable. The user application must ensure that all fields as reflected by the MSP DSECT are accounted for, either with or without data. The user is responsible for forward chaining on file the blocks of storage which will contain his data.
8. Two additional bytes are provided for each output field in the user's DSECT. The purpose of this is to provide the user application program with the capability to override the attribute byte defined in the Terminal Map Record at the time the output data is being created. The attribute byte override is simply moved into the left-most byte of the two byte field. It *must* contain a bit configuration recognizable by the 3270 hardware or the results will be unpredictable. *No* validation of user generated attributes is performed by the output mapping function.
9. When the user desires to output to shared buffer printers (3278/3284) there are various options that must be taken into consideration:
   a. For RCB dependent applications the RCB for the CRT will contain the address of the associated printer so applications should pass either the printer RCB or LNIATA.
   b. For systems with no RCBs there is only one RVT entry for the CRT/printer combination, so the user must enter output mapping with the WCC start print on and pass the resource ID (RID) of the combined unit.

c. There are no shared buffer terminals for the extended 3270 CRTs (3278). When the user wants to transmit to a CRT and a printer the WCC start print bit must be on. The cluster controller will display the message on the CRT specified by the RID and will print the message on the first printer it finds that is not busy.

10. Bytes 3–5 of the mapping interface work area enable the user to either override the 3270 command, WCC, and CCC in the Terminal Map Record or to provide the FROM address for a COPY command. When the user wishes to override the 3270 command, the one byte command (write, E/W, Copy, Erase All Unprot, Erase/Write Alternate) code is moved into *byte 3* of the work area. When a 3270 WCC or CCC is to be overridden, the desired eight bit character is moved into *byte 4* of the work area. When a COPY command has been specified, either in the Terminal Map Record or byte 3 of the work area, byte 5 must contain the 3270 address of the device being copied from. It is the user's responsibility to place the correct values and characters in bytes 3–5. If bytes 3, 4, and 8 are not used, they must be initialized to values of X'00'. Also, the user must ensure the unsolicited bit is set in the RCPL (RC0PL).

    The purpose of the following sample code is to provide an example of how the entry requirements to output mapping (ASFO) may be accomplished:

```
            BEGIN
             ---
             ---
             ---
             ---
            GETCC   D0,L2 OBTAIN 1055 BYTE BLOCK
            L       R2,CE1CR0                LOAD BASE ADDRESS
            MVI     MAP2OBID,X'00'           INITIALIZE BLOCK
            MVC     MAP2OBID+1(256),MAP2OBID
            MVC     MAP2OBID+256(256),MAP2OBID
            MVC     MAP2OBID+512(256),MAP2OBID
            MVC     MAP2OBID+768(256),MAP2OBID
            MVC     MAP2OBID+1024(26),MAP2OBID
            MVC     MAP2OLIT(4),LITC          LNIATA & CPUID
             ---
             ---
             ---
             ---
             ---
            XC      EBW000(18),EBW000
            LA      R1,#MAP2O                MAP SEQUENCE NBR
            STH     R1,EBW000                STORE IN WORK AREA
            MVI     EBW002,X'80'             INDICATE NO RETURN
            MVC     EBW006(12),PRCPL         MOVE RCPL TO WORK AREA
            LA      R1,EBW000
            ENTNC   ASFO                     OUTPUT MAPPING
             ---
             ---
             ---
             ---
             ---
LITC        DC      XL4'030802C1' LNIATA & CPUID
PRCPL       DC      XL12'Reference Routing Control Parameter List
                    RC0PL'
            FINIS
```

## Page/Scroll Entries

Mapping activates the Page/Scroll program (03-ASC) when more than one screen is needed to complete a display. The Page/Scroll program, in turn, creates a

Multiple Screen Control Record (AS5MB) for this entry. This allows the program to control each display, based on user input and information contained in the AS5MB record.

The user, at map creation time, can specify the type of display that best fits his needs. For information, see “Terminal Panel Display Macro (DPANL)” on page 71. If SCREEN=SPLIT was specified at map create time, the Page/Scroll program can interpret input for the upper portion of the screen, the lower portion or both. This allows the user to page or scroll the upper half or lower half of the screen independently of each other. If SCREEN=FULL was specified or the default taken, the Page/Scroll program accepts input only from the last line of the screen and pages or scrolls the entire display.

The following pages provide examples of messages required to invoke the page/scroll function. The page and scroll entries are shown with their respective responses.

## Example 1: Initial Page/Scroll Display

```
                DEN TUL         08/04/74
                        CLASS   FARE     AIRLINES

                          A     88.80    AA AC AD AE AF AG
                                         BA BC BD BE BF BJ
                          T     44.07    AB AC BE
                          F     88.80    AB AC AE AT AJ AK
                                         BB BC BE
                          Y     44.07    AA AC AD AE AF AH
                                         BA BH
                          Z     44.07    AA AC AD AE AF AH
                                         BA BC BD BE BF BH

Agent Request:    PD2


Where:  PD2 = Page forward 2 pages.
```

## Example 2: Response From Above Agent Request

```
                DAL       F     88.80    AA-AC AD-AE AF-AH
                                         BA-BC BD-BE BF-BH
                ELP       T     88.80    AA-AB AC-AD AE-AF
                                         BA-AB CA-BC
                          A     44.07    AB-AC CA-CD EF-AB
                                         BB-BC

Agent Request:  MU3

Where:  MU3 = Move display forward three lines.
```

## Example 3: Response From Above Agent Request

```
                                         BB BA BC BD BE BF

                        CLASS   FARE     AIRLINE PAIRS

                DAL       F     88.80    AA-AC AD-AE AF-AH
                                         BB-BC BD-BE BF-BH
                                         CA-CC CD-CE CF-CH
                ELP       T     88.80    AA-AB AC-AD AE-AF
                                         BA-AB CA-BC
                          A     44.07    AB-AC CA-CD EF-AB
                                         BB-BC
```

```
Agent Request:    PT

Where:  PT = Redisplay page 1.
```

## Example 4: Response To Agent ″PT″ Entry

```
                DEN TUL                 08/04/74
                        CLASS   FARE    AIRLINES

                          A     88.80   AA AC AD AE AF AG
                                        BA BC BD BE BF BJ
                          T     44.07   AB AC AE
                          F     88.80   AB AC AE AI AJ AK
                                        BB BC BE
                          Y     44.07   AA AC AD AE AF AH
                                        BA BH
                          Z     44.07   AA AC AD AE AF AH
                                        BA BC BD BE BF BH

Agent Request:  PD12B

Where:  PD12B = Incorrect page request.
```

## Example 5: Response To Above Agent Request

```
                DEN TUL                 08/04/74
                        CLASS   FARE    AIRLINES

                          A     88.80   AA AC AD AE AF AG
                                        BA BC BD BE BF BJ
                          T     44.07   AB AC AE
                          F     88.80   AB AC AE AI AJ AK
                                        BB BC BE
                          Y     44.07   AA AC AD AE AF AH
                                        BA BH
                          Z     44.07   AA AC AD AE AF AH
                                        BA BC BD BE BF BH

                 INVALID FORMAT
```

## Example 6: Initial Page/Scroll Display With Split Screen

```
                DEN TUL                 08/04/74
                        CLASS   FARE    AIRLINES

                          A     88.80   AA AC AD AE AF AG
                                        BA BC BD BE BF BJ
                          T     44.07   AB AC AE
                PB
                          F     88.80   AB AC AE AI AJ AK
                                        BB BC BE
                          Y     44.07   AA AC AD AE AF AH
                                        BA BH
                          Z     44.07   AA AC AD AE AF AH
Agent Request:   PD2

Where:  PB  = Page to last page.
        PD2 = Page down 2 pages.
```

## Example 7: Response To Above Agent Request

```
                                        AA-AC AB-AD AF-AE
                                        BB-BC
                          Y     44.07   AB-AC
```

```
                DAL       F     88.80    AA-AC AD-AE AF-AH
                                         BA-BC BD-BE BF-BH
                MU3
                                         CA-CC CD-CE CF-CH
                ELP       T     88.80    AA-AB AC-AD AE-AF
                                         BA-AB CA-BC
                          A     44.07    AB-AC CA-CD EF-AB
                                         BB-BC

Agent Request:   PU2

Where:  MU3 = Display 3 lines more from top.
        PU2 = Page up 2 pages.
```

### Example 8: Response To Above Agent Request

```
                HNL       F     88.80    AA-AB AC-AB AD-AC
                                         BC-BD
                          M     44.07    CA-CC CB-CD CE-CF
                                         AA-AC AB-AD AF-AE
                                         BB-BC
                ►
                          F     88.80    AB AC AE AI AJ AK
                                         BB BC BE
                          Y     44.07    AA AC AD AE AF AH
                                         BA BH
                          Z     44.07    AA AC AD AE AF AH

                 ►

Where:  ► indicates where user is to input page/scroll request
           (First character of page/scroll request overlays ►
           character)
```

## System Errors

A description of system error numbers issued by the TPF mapping support package can be found in *Messages (System Error and Offline)*.

# Message Switching

The message switching data records are to provide maximum flexibility in adapting the application to the system requirements. Provision is made to operate with many different types of circuits and terminals. However, the programs in the system are modular in design and only those functions required need be incorporated.

The effort to initialize a message switching system, having a specific hardware configuration, can be determined by defining the following areas:

- High speed lines
- Terminals - Line, IA/TA Assignments
- City/Airport Codes
- Office Function Codes
- Group Codes - Special Distribution Lists

After defining these areas, the following outline may be used to initialize the required data records.

Some consideration for expansion of the customer's configuration should initially be built into the tables. This would help prevent a regeneration of the tables at an early date.

## Records To Be Initialized

## Records Generated Off-Line

The following records are initialized by using the System Test Compiler to build a pilot tape. The pilot tape is subsequently loaded onto the system files by the System Data Loader. For more information about the System Test Compiler, see *TPF Operations*.

Fixed-file records are written to their allocated address and are retrieved by Message Switching programs by using File Address Compute.

Core Resident records are loaded into the Global Area by the Application Core Load and Restart Program at system restart time. These records are then accessed by Message Switching programs by using the appropriate tag in the Message Switching Global Field.

### Records Requiring Initialization with Control Information and Data Fields

| | |
|---|---|
| **XC1CC** | City Code Index Table |
| **XK1CT** | Terminal Index |
| **XF1FF** | Function Code Index Table |
| **XI0DS** | Input Log Directory (Optional) |
| **XX1ON** | Line IA/TA Index |
| **XJ1LC** | Line Index |
| **XS0AA** | XLMT Assembly Area |
| **XW1OC** | Output Control Table |

**XL0DS** Output Log Directory

**XP1XP** Physical Line Conversion Table

**XB0XB, XB1XB**
Statistical Annex Table

**XD0LS** Special Distribution Lists

**XZ1AT** Terminal Annex Table

**XR1TR** Terminal Routing Table

### Records Requiring Initialization With Control Information Only

**XV1XV** Input Control Table

**XN1XN** Input Status Table

**XE1SC** Message Segment Count Table

**UV1RP** Retrieval Table

**UV3RP** Purge Table

**XT0CB** Transmission Control Block

**XQ1XQ** Terminal Intercept Queue

**XU2TQ** Transmission Queue Control Tables

## Records Generated By Message Switching Programs

The following records are initialized by Message Switching programs as they are required. The file pool records are released after they are used.

**UW2CP** Checkpoint Chain

**UQ1UQ** Error Back-up Copy Records

**XZ9ER** Error Correction Work Area - level six

**XM0RL** Communication Message Format

**UZ1PQ** Process Queue

**UR1DS** Station Designator Sequence Table

**XH0XH** Terminal Intercept Block

**UU1TT** Terminal Time Table

# Items To Be Initialized within the Data Records

## Records Requiring Initialization with Control Information and Data Fields

The following describes the fields that must be initialized to some specific value within each data record.

### City Code Index Table

Core resident record - Core address in Global @X1CIC.

One entry (5 bytes) per City/Airport designator or group code serviced by the System (3 bytes) and cumulative office function count (2 bytes).

File Copy resides in 1055 byte record(s).

Fields to Initialize:

**XC1CC** Record Label

**STDHDR** ID = XC, Length = 1055, no chaining

**XC1TSZ** Number of City Code entries in the table (2 bytes - Bin.)

**XC1CAD** City/Airport or Group Code (3 bytes BCD)

***XC1CUT** Cumulative Function Count (2 bytes - Bin.)

* Above data for each entry (up to 208) - last entry will be blank.

**Method:** Accomplished when City/Airport and Office Function codes are known.

## Terminal Index

Core resident record - Core address in Global @X1CTI.

One entry (3 bytes) per IA/TA.

File Copy resides in 1055 byte record(s).

Fields to initialize:

**XK1CT** Record Label

**STDHDR** ID = XK Length = Number of entries, no chaining

**XK1CTI** One item entry (3 bytes) defined below

**XK1TSX** 2 bytes Bin - IA/TA for High Speed Terminal.

**XK1TON** 1 byte Bin XTRT Terminal Ordinal Number (TON, 0-255)

**Method:** Accomplished when IA/TAs are assigned for terminals.

## Function Code Index Table

Core resident record - Core address in Global @X1FIC.

One entry (3 bytes) per office function designator (2 bytes) and terminal ordinal number (1 byte).

File copy resides in 1055 byte record(s).

Fields to Initialize:

**XF1FF** Record Label

**STDHDR** ID = XF, Length = 1055. No chaining.

***XF10FC** Function Code (2 Bytes BCD).

***XF1OTN** XTRT Terminal Ordinal Number (1 byte) or Special Distribution List Ordinal Number

* Up to 348 additional entries, same format.

**Method:** Accomplished when function codes are assigned to terminals, or to Special Distribution Lists in the case of group assignments.

## Input Log Directory (Optional)

381-byte fixed file record.

23 records per terminal, circular chain with last chained to first.

Fields to Initialize:

**XIODS** Record Label

**STDHDR** ID = XI, forward chained and last to first.

**XIOESZ** Entry Size = 8 (X'0008'). Balance of record initialized to zero.

**Method:** Initialized if input logging is specified.

## Line IA/TA Index

Core Resident Record - Address in Global @X1LAD.

One entry (4 bytes) for each terminal - assigned at System Assembly time.

The first entry is reserved as the Prime CRAS entry (Line - 01, IA - 00, TA - 00). The second entry is reserved for Receive Only (R.O.) CRAS (Line - 01, IA - 00, TA - 02).

File copy resides in 1055 byte record(s).

Fields to Initialize:

**XC10N** Record Label

**STDHDR** ID = XX, Length = 1055, no chaining

**XX1ELN** Entry Length = 4

**XX1TLN** 2 bytes Bin. Table Length. Count of table entries - 1.

**XX1TEN** Terminal entry 4 bytes as follow

**XX1LNR** Symbolic Line Number (LN) (1 Byte Bin.)

**XX1TIA** Terminal Interchange Address (IA) (1 byte Bin.)

**XX1TTA** Terminal Set Address (TA) (1 byte Bin.)

**XX1IN1** (1 byte) Initialized to zero

Additional entries (up to a maximum of 256 entries) follow same format.

**Method:** The LN, IA and TA are in System/370, 8-bit code format arranged in an ascending numerical order. Entry ordinal numbers (which correspond to the associated XLMA) are assigned as a result of the entry's position in the XLAD table.

## Line Index

Core Resident Table - Address in Global @X1LCI. One entry (2 bytes) per line assigned to the System.

File copy resides in 1055 byte record(s).

Fields to Initialize:

**XJ1LC** Record Label

**STDHDR** ID = XJ, No chaining

**XJ1LCI** Cumulative IA/TA Count (2 bytes)

**Method:** Accomplished when IA/TAs are assigned to system lines.

## XLMT Assembly Area

381-byte record located in fixed locations on file.

One record reserved for each hard copy terminal assigned to the system, ordered sequentially to correspond to each entry in the XLAD table.

Fields to Initialize:

**XS0AA** Record Label

**STDHDR** ID = XS Length = 381, no chaining

**XS0OLN** Symbolic Line Number (1 byte)

**XS0TIA** Terminal Interchange Address (1 byte)

**XS0TTA** Terminal Set Address (1 byte)

**XS0TTP** Type of Terminal (1 byte)

**XS0MTL** Output Message Transmission Length (1 byte)

**XS0OTM** Transmission Method (1 byte)

**XS0TOQ** Type of Output Queue (1 byte)

**XS0CAP** Output capacity of XLMT Assembly Area (2 bytes)

**XS0OTL** Output Transmission Length (2 bytes)

**XS0TQC** Message Switching XTRT Terminal Ordinal Number (XR1TR), (2 bytes)

**XS0TON** XLMA Ordinal Number (2 bytes)

**Method:** The above fields are initialized to define the terminal type and transmission characteristics when the types and addresses of hard copy terminals are defined.

## Output Control Table

Core resident table - address in Global @X1OCT.

One entry (12 bytes) for each high speed line. Every 40th entry is unused, when headers on 1055 byte tables following the first are included.

File copy resides in 1055 byte record(s).

Fields to Initialize:

**XW1OC** Record Label

**STDHDR** ID = XW Length = 1055 No chaining

**XW1ESZ** Entry Size (2 bytes Bin.) currently = 12

**XW1LNR** (1 byte Bin.) Line number of physical output line

**XW1IN1** (1 byte) bits 0-2 set for terminal type.

**Method:** Initialized when system lines are defined.

## Output Log Directory

381-byte record located in fixed file storage.

23 records required for each terminal in the System. Forward chained only, with 23rd record chained to first.

Field to Initialize:

**XL0DS** Record Label

**STDHDR** ID = XL, Size = 381, forward chained only (last chained to first).

**XL0ESZ** (2 bytes Bin.) Entry Size = 8. Balance of record initialized to zeros.

**Method:** Initialized when terminals are defined.

## Physical Line Conversion Table

Core Resident Table - Address in Global @X1PLC.

One entry (1 byte) for each ordinal line number related to each symbolic line number.

File copy resides in a 1055 byte record.

Fields to Initialize:

**XP1XP** Record Label

**STDHDR** ID = XD, Length = 1055, no chaining

**XP1CNT** (2 bytes Bin.) Number of entries in this table

**XP1LNR** (1 byte Bin.) Ordinal Line Number

**Method:** When system lines are defined, each is assigned a symbolic line number.

## Statistical Annex Table

381-byte fixed file records.

One entry (52 bytes) per line with maximum of 7 entries per block.

Minimum 4 blocks - one for each type report.

The first four XSAT records are initialized to contain constant report information (XB0XB) - title, date and column headings for each report. Character Count, output line number and Record ID are also inserted. Station Designator of terminal designated to receive reports is inserted in the first record. Initially, any records after the fourth need only have the ID and entry size entered.

**XB0XB and XB1XB** Record Labels

**STDHDR** ID = XB

## Special Distribution Lists

381-byte fixed file record.

Each entry size is equal to as many bytes as required to provide one bit for each terminal ordinal number in the system.

Fields to Initialize:

**XD0LS** Record Label

**STDHDR** ID = XD, Size = 381, no chaining

**XD0ESZ** Entry Size = 1 bit for each T.O.N. in system

**XD0LST** List Start

**Method:** Initialized when special distribution codes are defined.

### Terminal Annex Table
381-byte fixed file records.

One entry (36 bytes) per XTRT terminal - 10 terminals per record.

**STDHDR** ID = XZ, forward chaining, length 381. Label- XZ1AT

**XZ1OCT** XOCT Ordinal Number (must correspond with output line for this terminal). X'FF' placed in XZ1OCT of unused entries.

**Method:** Initialized when message switching terminals are defined.

### Terminal Routing Table
Core resident record - address in Global @X1TRT.

One entry (12 bytes) for each terminal attached to the system.

The first entry is reserved for Airlines Reservations. This entry is followed by High-speed terminal entries.

File copy resides in 1055 byte record(s).

Fields to Initialize:

**XR1TR** Record Label

**STDHDR** ID = XR Length = 1055, no chaining

**XR1TSZ** Entry count (total number of entries in table)

**XR1ESZ** Entry Size = 12

***XR1RSD** Receiving Station Designator (3 bytes)

***XR1LNN** XTQC Ordinal Number (1 byte)

***XR1CDC** XLMA ordinal number for HS terminal

***XR1IN2** Terminal type

* Maximum of 86 entries per record have same data format.

**Method:** Initialized when message switching terminals are defined.

## Records Requiring Initialization With Control Information Only

The following records are initialized with the Record ID and control information only and the data areas are to be set to zero.

### Input Control Table
381-byte fixed file record. 20 Lines/Terminals per record.

**STDHDR** ID = XV, Length = 381, no chaining Label - XV1XV

### Input Status Table
Fixed Core Resident Record - Address in Global @XN1XN.

File copy in 1055-byte record is used for file back-up.

Byte requirement is number of HS Terminals times 4 bytes per terminal.

**STDHDR** ID = XN, Length = 1055, no chaining, Label - XN1XN

**XN1ESZ** Entry Size = 4

### Message Segment Count Table
Fixed Core Resident Record - Address in Global @XE1SC.

One byte for each high speed terminal.

File copy in 1055 byte record for back-up.

**STDHDR** ID = XE, Length 1055, no chaining, Label - XE1SC

### Retrieval Table
381-byte fixed file record. File address in Global @XJRPT.

Stored in the file area reserved for retrieval-purge tables and assigned an ordinal number of one.

**STDHDR** ID = UV, Size = 381, no chaining, record code check = X'01', Label - UX1RP

**UV1RDN** Maximum Entry Count (maximum number of entries in XRBQ, XRDQ, XRPQ records).

### Purge Table
381-byte fixed file record. File address in Global @XIRPT.

Stored in the fixed file area assigned to retrieval-purge tables and assigned an ordinal number of zero.

**STDHDR** ID = UV, size = 381, no chaining, Label - UV3RP

### Retrieval Reel Table
381-byte fixed file records.

32 records required - one plus one for each day of the month.

**STDHDR** ID = UT, Size 381, no chaining, Label - UT2RT

**UT2EPD** No. of entries per day = 10

**UT2ESZ** Size of entry = 12 bytes

### Transmission Control Block
381-byte fixed file primary record for each XTCB queue. Three XTCB queues (one per priority) are maintained for each HS (1977) message switching terminal. Initialized by loading empty XTCB records into file addresses obtained from FACE using record type #XTORI. Overflow records are obtained from the file pool.

**STDHDR** ID = XT, Control byte XT0CTL Set, forward chaining - Size = 381, Record Code Check (RCC) = queue ordinal number, Label - XT0CB.

**XT0ESZ** Entry Size - Set to 8 bytes

### Transmission Queue Control Tables
Fixed core resident record. Address in Global @X1TQC. One entry (26 bytes) per HS Terminal.

39 entries/record.

Ordinal entries 39, 79, 119, 159, 199, and 239 are invalid if header entries are not deleted in core table records after the first XTQC record. (Refer to Transmission Queue Control Tables.)

**STDHDR** ID = XU, Size = 1055, no chaining. Label - XU2TQ.

**XU2CB1** Last file address - Initialized equated to prime XTCB record.

**XU2ESZ** Entry Size = 26 bytes

**XU2IN1** Status Indicator - Set to X'80' - Queue empty priority 1.

**XU2IN2** Status Indicator - Set to X'80' - Queue empty priority 2.

**XU2IN3** Status Indicator - Set to X'80' - Queue empty priority 3.

**XU2PSZ** Priority Size = 8 bytes

## Records Generated By Message Switching Programs

The following records are short-term records secured from the available file storage pool and initialized by the user program. A brief description is made to be aware of their usage and existence.

### Check Point Chain

Label - UW2CP, Size = 381, forward chaining. Used by the Periodic Purge as a temporary copy of the Output Log Directory (XL0DS).

### Error Back-up Copy Records

381-byte fixed file records. 80 records - for XTAT and XSAT copies. Add one record for each XLMA record. XCPY record label - UQ1UQ. The records require no special initialization other than proper formatting of the file. XCPY records are initialized by the message switching restart program each time the originals are reinitialized.

### Error Correction Work Area

381-byte core record.

Temporary reference record maintained in core by the Application's Message Switching Error Correction Program.

### Communication Message Format

381-byte file record random pool. ID = TM, forward chaining.

One record is required for each message segment stored in the system.

Created by the Input Message Assembler Program (03-XIMA) and formatted according to XMSG specifications.

The backward chain field of the primary record contains the file address of the last record of each multi-record message.

### Process Queue

381-byte file pool record.

One record - maximum 30 entries at 12 bytes per entry.

**STDHDR** ID = UZ, Size = 381, no chaining. Label - UZ1PQ

**UZ1ESZ** Entry Size = 12 bytes

### Station Designator Sequence

1055-byte file pool record. Label - UR1DS.

Initialized by the Purge Program when it generates this table. Written to purge tape when created.

### Terminal Intercept Block
381-byte file pool records.

Each record has a capacity of 45 XTIB entries (low speed intercepted message references). The number of records required is a function of the number of terminals and priorities on intercept. If average of 5 terminals on intercept, 12XTIB records would be required if a maximum of 540 messages are held. As long as terminals remain on intercept, XTIB records are necessary.

Used exclusively by the Intercept Program and Retrieval Program to queue a message of a terminal and priority on intercept and subsequently return the message to the traffic main stream on retrieval of intercept.

### Terminal Time Table
1055-byte core record.

Created by the Purge Program with first and last sequence numbers and a list of the date-time groups of these messages for each terminal. Written to tape preceding the messages purged for a particular terminal. There is one XTTT for each terminal purged to tape.

## Message Switching Global Entries

Those message switching entries needed to define the configuration used by the customer reside in the Global area.

Following is a list of all message switching global entries, the value to which they are initialized and an explanation. Expansion should be included to reduce the amount of work to regenerate values.

| | | | |
|---|---|---|---|
| @XIPLG | DC | X | Switch for input log or not<br>**X'00'** for no input logging<br>**X'FF'** if input logging desired |
| @XIPCT | DC | H | Number of terminals not assigned to input log |
| @XSATS | DC | H‘7’ | Number of entries per XSAT record. |
| @XTATS | DC | H‘10’ | Number of entries per XZ1AT record. |
| @XSYST | DC | X | Type of System<br>**X'00'** PARS only<br>**X'80'** MS only<br>**X'01'** PARS and low-speed MS<br>**X'02'** PARS and high-speed MS<br>**X'03'** PARS and mixed (low and high-speed) MS |
| @XSYTC | DC | H | Number of entries in XR1TR (One plus the number of high-speed MS terminals). |
| @XSYLC | DC | H | Number of MS lines per system (Sum of high-speed MS lines). |

| | | | |
|---|---|---|---|
| @XMXML | DC | H | Maximum number of segments (381 byte blocks) per MS message. |
| @XMXRT | DC | H | Maximum time range for a retrieval request before it is sent to CCS for approval. |
| @XMXRC | DC | H | Maximum number of messages to be retrieved before being sent to CCS for approval. |
| @XKIND | DC | X'00' | Message switching switch (used by MS programs and Recoup). |
| @X1SDL | DC | H | Number of entries per XD0LS record. (Value derived from 360 divided by the rounded up @XSYTC) 8 |
| @XMAVS | DC | H | Mask for AVS/ASC should correspond to XTRT ordinal number for AVS/ASC line. |
| @XNHST | DC | H | Number of high-speed terminal entries in XIST. (XICT should correspond). |

The following two fields require field reservation in the global area, however, they are both initialized by the MS Restart Program.

| | | | |
|---|---|---|---|
| @XIRPT | DC | F | Retrieval Table (UV1RP) file address. |
| @XJRPT | DC | F | Purge Table (UV3RP) file address. |

## Core Table Headers Categories

Message switching core tables are loaded from 1055 byte file records. The amount of core required per table will determine how many file records are needed. Because of the programming techniques used in handling these core tables, the tables are divided into three categories.

## Single File Record Core Table

These message switching core tables are composed from only ***one*** file record because of built-in limits.

XN1XN
XJ1LC
XE1SC
XP1XP

## Multiple File Record Core Tables - Header Dropped

This group of core table records are loaded with headers dropped from all file records but the first record of that core table. The reason for this is that all data must be accessed from adjacent core locations. Therefore, the second record must be loaded consecutively into core after the first, etc.

XC1CC
XK1CT
XF1FF
XX1ON

## Multiple File Record Core Tables - Header Not Dropped

The following core tables have provisions to skip header entries of other than the first file record comprising that core table. The file records must, however, be loaded consecutively one after the other.

XW1OC*
XY0XY*
XR1TR**

***** It should be noted that the XW1OC and XU2TQ can also be placed in Category B (Header Dropped). The XR1TR cannot be placed in Category B because it is keypointed from core.

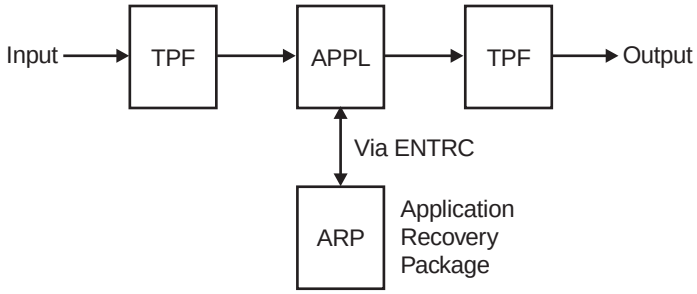
****** XR1TR is keypointed from core. This is done to preserve indexes and status indicators over a restart.

# Application Recovery Package

Application recovery support is an option of the TPF system environment. This package uses the application recovery table, AR0RT, as a vehicle for providing a user (application) with message recovery capability.

The integrity of the application recovery table (ART) is maintained by the system; this package provides the application with the means by which an ART item can be obtained, referenced, keypointed, and deleted. At any point in its processing and cycle, an application can save input, output, or user data on file and reference it via its associated ART item, and/or it can save data in the optional data area of the ART item. An ART item, once obtained by an application, will be maintained until deleted by the application. Should application processing be interrupted because of application or system error, this package provides a *timeout* function which will reactivate the application using Routing Control data (RC0PL) maintained in the ART item. The application interface program, referenced in the Routing Control Table, cannot be corefast.

The following diagram illustrates the area in which application recovery operates:




## Description

### Purpose

This package is a tool which the user (application) can use to supplement existing system message recovery capability an/or to combine message recovery function with data base recovery function. This type of implementation can lead to shortened transaction path length and/or improved data integrity. The application can choose the point of recovery, the location and contents of log records, if any, and the tracking (timeout) period. This function is independent of line disciplines, and, thus, can be used with BSC or SNA devices. The user determines the extent of and provides for this recovery capability. The application recovery (AR) package provides the interface to a system maintained data area.

### Environment

When system generated, this package enables an application (user) to assume responsibility for all non-SNA input and output message recovery. If system SNA input recovery is not system generated, then the package enables the application to assume responsibility for SNA input message recovery as well. If system SNA input recovery is system generated, then the TPF system will be responsible for SNA input recovery. SNA output recovery is the responsibility of the the TPF system. Any data base recovery function is the responsibility of the application.

The application recovery table (ART) resides in the protected global area and is loaded by GOGO (INSL-GLBL, Application Global Area PLM). The Core Allocator Record (INSL-GLBL) will have been updated to reserve the required amount of core for the ART table. The GOA is used by GOGO to load the global area.

## Functions

- To perform application requests to obtain, locate, and delete an application recovery table slot reference and file application data on request.
- To initialize the ART core table and delete slots with no timeout during Restart.
- To keypoint the ART core record on a time-initiated basis. The entire core copy will be written to file when requested by the AR package and individual blocks will be written when user requested.
- A timeout program to cycle through the ART core table and reactivate the application with the RCPL contained in the ART slot, when timeout occurs.

## Package Program List

**APPLICATION RECOVERY PROGRAM** (03-ARP)

This program, activated by user request and the Restart function, controls access to and maintenance of the application recovery table (ART).

## Relationships To Other Components

### Programs

- **APPLICATION MAIN STORAGE LOAD PROGRAM** (INSL-GLBL)

  This program is used to load the core application recovery table (AR0RT) to the global area.
- **CPS SCHEDULES**

  This program activates the application recovery initialization program during restart.
- **CALCULATE UNIQUE NUMBER** (03-CVN)

  This program supports the Random Number Table (CVNH) and the *hashing* algorithm used by the AR Package to access a unique slot in the ART. This interface is through the HASHC macro.

### Records

- **MAIN STORAGE ALLOCATION RECORD** (GO1GO)

  Items within this record define the amount of core necessary to reserve for the ART and the location within the global area. This and other information is used by GOGO to load the ART.
- **APPLICATION RECOVERY TABLE** (AR0RT)

  Core record maintained by the application recovery Program and used by an application to preserve information about a particular input/output message/function.
- **ROUTING CONTROL PARAMETER LIST FORMAT** (RC0PL)

  Block associated with each TPF input/output message to provide information about the origin, destination, and characteristics.
- **KEYPOINT RECORD A** (CK1KE)

The timer value used to activate the timeout program (ARPT) is stored in this record. This program record is retrieved and the timer value (CK1ART) is used for the CRETC to activate ARPT. This value is SIP (System Installation Package) input via RCYTO in macro MSGRT.

- **ROUTING CONTROL APPLICATION TABLE** (RC0AT)

  When timeout occurs, the application name in the RCPL is used to find the associated RCAT entry so the appropriate application can be reactivated.

## Input Action

Functions available for application activation are: obtain, locate, file and delete an ART slot.

Parameter List Format:

This consists of consecutive fullwords starting on a fullword boundary, each fullword containing an address to be passed. Register 1 (R1) contains the address of the start of the parameter list:

| Word | Contents |
|---|---|
| 0 | Address of user supplied RCPL (RC0PL)<br>OR<br>Address of core slot (AR0RT) |
| 1 | Address of input area. |

This input area consists of:

Byte 0- Request Indicator (bit indicator)

The following equates are used to set the request indicator:

| | |
|---|---|
| **SET = AR0OBTN** | OBTAIN A SLOT |
| **SET = AR0DLET** | DELETE A SLOT |
| **SET = AR0FILE** | FILE MESSAGE REQUEST |
| **SET = AR0LOCA** | LOCATE A SLOT |

Byte 1- Level to File from for File Request

The following equates are used to indicate the level:

| | |
|---|---|
| **SET = D0** | for level 0 |
| **SET = D1** | for level 1 |
| **SET = D2** | for level 2 |
| **SET = D3** | for level 3 |
| **SET = D4** | for level 4 |
| **SET = D5** | for level 5 |
| **SET = D6** | for level 6 |

Byte 2-5 - Available for return of core address of ART slot.

Byte 6-17 - Work area necessary for processing delete request only.

## CPU Process

## ART Functions

This program makes available, on application request, an ART slot for modification. During application activity, the slot information may be re-accessed if the core location is lost. On completion of the application processing of this message, the slot reference may be deleted with indication that this slot is now available for re-use.

Initialization is based on inputs, supplied by PILOT load, in the file copy of the application recovery table (ART). At this time, the number of items and the length of the variable portion of an item (AR0LEN= length of AR0DTA) are supplied. This information is used after GOGO load by the application recovery program to initialize the ART in core. Inputs are then set up to activate the timeout program. The value used for the CRETC is picked up from Keypoint Record A (CTKA, CK1ART).

During system restart, if the user has requested no keypointing, the ART is re-initialized and all slots set to available. When keypointing is valid, the ART table is searched. Any slots containing no timeout value are zeroed and that slot is made available for re-use. This is a maintenance function to free up slots that the user has lost access to due to restart and user request for no timeout. Upon completion, the updated core table is copied back to file.

Timeout is activated at user specified time intervals. This function cycles through the ART locating items in use that have a timeout value. This field is decremented by 1. If timeout occurs, the RCPL is routed back to the origin. At completion of this function, this program CRETs to itself, again using the timer value from Keypoint Record A, to ensure re-activation.

The keypoint facility is activated on a time-initiated basis. This means writing the core information contained in the ART back to file to preserve the application data. This may be on a per record basis set by the user (slot indicator AR0SY1=AR0KYPT) or for the entire core copy set by the AR package (AR0IND=AR0KEYP).

## Outputs

Message to console after GOGO global core load:

```
'ART LOADED'
```

## Output From ART Function On Return To Application

```
ECB Work Area -

  EBCM01          Return Code

Parameter List Format:
```

This consists of consecutive fullwords starting on a fullword boundary, each fullword containing an address to be passed. Register 1 (R1) contains the address of the start of the parameter list:

| Word | Contents |
|---|---|
| 0 | Address of user supplied RCPL (RC0PL)<br>OR<br>Address of core slot (AR0RT) |
| 1 | Address of input area.<br><br>This input area consists of:<br><br>**Byte 0**<br>Request Indicator (bit indicator)<br>The indicator is turned off on return.<br><br>**Byte 1**<br>Level to File from for File Request<br>The indicator is unchanged.<br><br>**Byte 2-5** Core address of ART slot.<br><br>**Byte 6-17** Unpredictable data. |

## Output To Application From Timeout Function

User RCPL (Routing Control Parameter List Format, RC0PL) stored in the entry control block. The RCPL indicator identifies if this is a duplicate input message or returned output.

# Mapping Support Package

The mapping support package enables the user to format and control data streams routed between a terminal and an application program, and between two application programs. As a formatter vehicle, this package presents to the user program data streams devoid of device dependent code. This formatting capability is attained by user defined and package generated formats, or maps, which characterize input and output for CRTs and output for printer devices. The mapping support package additionally provides paging and scrolling multiple screen management control functions for CRT type devices.

## Description

### Purpose

The purpose of the mapping support package is to construct and format application to application, application to terminal, and terminal to application data streams. Input messages originating with field oriented terminals (e.g., 3277 display station) consist of device dependent control characters and text. This package deletes the former and arranges the latter in a symbolically addressable manner for the requesting user program. Similarly, output data streams created by user application programs are comprised solely of text. This package inserts into this data stream the appropriate special device control characters required to display the information on field oriented terminals (e.g., 3270 CRTs) or to print the formatted information on hard copy devices (1980-24, 3284, 3286, 3287, 1977).

### Functions

- To create symbolic reference modules based upon user defined formats and field definitions.
- To initialize the Terminal Map Record Files and to add, delete, or replace individual records in these files.
- To delete terminal dependent control characters from input message streams and format the remaining portion of the message in a manner symbolically addressable by a user application program.
- To insert the appropriate terminal dependent control characters into an output message stream created by a user application program.
- To provide control of multiple screen data displays for field oriented 3270 CRT terminals.
- To convert input data to upper case if desired.

### Component Programs

**Map File Create And Load Program**

This program, utilizing user created formats and field definitions, generates the Terminal Map Record files in an offline mode. These files, which consist of references to records that are to be deleted, replaced, and added, are then processed for loading onto the online files.

**Input/Output Mapping Program**

This program, which is activated by user request, maps or formats application to application, terminal to application, and application to terminal data streams. When mapping terminal to application or

application to application data streams, this program deletes device dependent control characters from the input and places the remaining data in a formatted message block corresponding to a DSECT which has been assembled into the user program. To perform this function, this program references information contained in the Terminal Map Record. The mapping of input data streams is available to field oriented terminals; currently this includes the 3277 Display Stations, Models 1 and 2.

When mapping application to terminal and application to application data streams, 03-ASF inserts device dependent special control characters into the user created text. The Terminal Map Record which defines the format of the message and field position and characteristics, is referenced to provide the necessary information to create the updated data stream. The output mapping facility is available for formatting 3270 display stations and printers, and the 1977, 1980-21, and 1980-24 hard copy terminals.

**Multiple Screen Control Program**

The first function of this program is to segment long messages that are to be displayed on a 3270 CRT into screen size elements or pages. The information defining the start of each page (AMSG file address and displacement) is stored in a Page/Scroll Control Record (AS5MB). Upon terminal operator request, this program determines the specific line number or page to be displayed from the definition in the MSS Record and constructs the appropriate screen size display for the terminal.

The second function of this program is to assemble input message blocks containing fields which have been modified by terminal operator action; these fields are components of a multiple page display at a CRT terminal. As the terminal operator completes data entry for each page or panel, 03-ASC files the input message and stores the prime file address in the MSS. When the terminal operator has completed multiple screen input activity, this program will assemble the chain of input blocks for mapping.

## Associated Programs

- System Message Processor
- File Address Compute
- Read AAA/RCB.

## Other Components of This Package

The generation of physical map records on file and symbolic reference modules (DSECTS) in assembled programs is dependent upon an input source of parameterized macro statements. The user defines and names fields and the characteristics associated with each format. These source statements are then catalogued under unique format or map names in a Map Source Statement Library; this library provides the input to the offline element of the Map File Create And Load Program (03-ASL) and the MVS Assembler.

**DPANL** Terminal Panel Display Macro

This macro provides the capability for the user to identify various displays or reports by assigning unique map names and map sequence numbers to each of the formats. In addition, the user

defines overall characteristics (terminal type, message block size, mode, upper case conversion for input data, etc.) for each of the formats.

**DDATA** Terminal Data Field Display Macro

This macro provides the capability for the user to assign symbolic names to individual fields and define those fields with various characteristics (length, 3270 attributes, constant text). The former allows the user program to symbolically address individual fields in a DSECT; the latter allows the user to vary the delineation of fields within a format.

## Relationship to Other Packages

All programs requesting mapping of terminal input/output and application to application data streams use this package.

## External Input and Output

## Agent Set Input/Output

```
Input:          SCROLL REQUEST

                MUnnn+
                MDnnn+
                MU+
                MD+
```

The primary action code for a terminal initiated scrolling request = M. The secondary option codes U, D, allow the terminal operator to move the display up (U) or down (D) a specified number of lines. The maximum number of lines that can be specified with a MD or MU request is 999. The absence of a numeric value implies one line only.

```
Output:         SCROLL REQUEST
```

The output will consist of the requested scroll segment or one of the following messages:

```
                INVALID REQUEST
                INVALID FORMAT

Input:          PAGING REQUEST

                PUnnn+
                PDnnn+
                PT+
                PB+
                PU+
                PD+
```

The primary action code for a terminal initiated paging request is P. The secondary option codes U, D, B, T allow the terminal operator to move the display up (U) or down (D) a specified number of pages, or to move to the first page of a display (T) or to the last page of a display (B). The maximum number of pages or screen panels that can be specified with a single PD or PU request is 101. Absence of a numeric value implies one page only.

```
Output:         PAGING REQUEST
```

The output will consist of the requested page display or one of the following messages:

```
                INVALID REQUEST
                INVALID FORMAT

Input:          MAP FILE LOAD/UPDATE

                ZAMAP+
```

This command, which initiates the map file load and update phase of this package, must be entered from a CRAS terminal only.

```
Input:          MAP FILE LOAD/UPDATE
```

When the map record files have been updated or loaded, the following response is output to the initiating terminal:

```
                MAINT COMPLETE
```

If errors have occurred during the processing sequence, the message(s) will appear on the RO CRAS device.

## Tape Input/Output

### Offline Terminal Map File Creation

The primary input source for the creation of map record items is the map source statements residing in the user created Map Source Statement Library. Control information is provided on card input and indicates whether processing is to be in an update or a create mode. If the former, card input will supply the information indicating whether the update is of an add, delete, or replace type. The offline segment of the Map File Create And Load Program (03-ASL) constructs the Terminal Map Records, which are sorted using the Sort Utility, and written to the Map Macro Tape File (AS0MP).

### Online Terminal Map File Load

The source of input for this processing phase is the Map Macro Tape File created offline. The information provided on this tape indicates whether processing is to be a complete file create/replace or a partial update of existing files.

## Other Input/Output

Mapping of data streams is initiated by a requesting user program. The user creates a map parameter list, which includes the selected map sequence number and 3270 command and control characters, as input to this package. After the data stream has been mapped and formatted, this package either returns the formatted data to the user or routes the formatted data to a user specified destination.

# Message Routing Package

This chapter describes the function and procedures that comprise the TPF message routing facility. The chapter is intended for system programmers, application programmers, and system operation personnel who are familiar with the TPF system as described in *TPF Concepts and Structures*. It is recommended that you review the information in *TPF Concepts and Structures* about the message router before using the information in this book.

SNA cross domain management and TPF message routing are related and occasionally entangled. Generally, TPF message routing implies that software in a CPU must be invoked to select a path for reaching a TPF application. The term *application* is always used in this chapter to indicate a class of related programs needed to accomplish a specific function, Coupon Remittance, Hotel Reservation, Credit Verification, Car Rental, and so on. In this context, a TPF application is defined as a software package that is reached through an entry in the Routing Control Application Table (RCAT). Complexity arises when the path to an application or terminal includes more than one CPU. This could involve multiple communications controllers or a single controller shared among several CPUs. The following information identifies some of the complexities of this topic. It is recommended that you review the information in *TPF Concepts and Structures*.

- Log processing to establish a connection between a terminal and a TPF application, within the SNA support, uses session services of the SSCP and path control function distributed to the Network Control Program (NCP) in a communications controller. The data messages following a *log in* message need not enter the host CPU of the terminal in order to reach an application in a remote CPU.
- Communication between two TPF applications in two CPUs is accomplished with the TPF ROUTC macro. Within an environment containing the Advanced Communication Function (ACF), the data link protocol must be SDLC. The path is selected by function distributed to the NCP, but the end points are determined by information in the SNA communication tables.
- Multidrop Binary Synchronous Communication (BSC) stations, as supported by the TPF system are viewed as intelligent nodes, that is, other CPUs. The multidrop BSC stations may be addressed, through the ROUTC macro, by using 4 character names which are held in a BSC Station Name Conversion Table (SNCT). A BSC station name implies different processing than is implied with a TPF application name, that is, a name in the Application Name Table (ANT).

Applications resident in the loosely coupled complex are defined to be resident in each of the LC processors. Therefore, if the loosely coupled complex consists of CPUs A and B, an application defined as locally resident in CPU A is also defined as locally resident in CPU B. The only exception to this is the System Message Processor (SMP) application. Viewed from CPU A, SMPA is locally resident and SMPB is resident in CPU B.

The Multiple Data Base Facility (MDBF) has minimal functional impact on the message router support. An application definition requires an identifier to associate the application with a Subsystem User name.

## Supported Devices And Protocols

IBM terminals supported on ALC lines by means of the ALCI feature of NCP are:

**2915** 12 line screen

**4505** 12, 15 and 30 line screens

**1977**

**1980**

Synchronous Data Link Control (SDLC) *terminals* are supported by the TPF system as System Network Architecture Logical Units (SNA LUs).

The SNA support communicates with the following SNA devices through the IBM 37x5 Network Control Program (ACF/NCP).

The 3601 and 3602 Finance Communication Controllers support the following devices:

| | |
|---|---|
| **3604** | Keyboard Display |
| **3606/3608** | Financial Services Terminals |
| **3610** | Document Printer |
| **3611** | Passbook Printer |
| **3612** | Passbook and Document Printer |
| **3618** | Administrative Line Printer |

The support for the IBM 3270 Information Display System consists of:

| | | |
|---|---|---|
| 3271 | Control Unit | Models 11 and 12 |
| 3274 | Control Unit | Model 1C |
| 3277 | Display Station | Models 1 and 2 |
| 3276 | Control Unit | Models 11, 12, 13 and 14 |
| 3278 | Display Station | Model 1 (960 characters)<br>Model 2 (1920 characters)<br>Model 3 in default mode (1920 characters)<br>Model 4 in default mode (1920 characters) |
| 3284 | Printer | Models 1 and 2 |
| 3287 | Printer | 2K or 4K print buffer |
| 3289 | Printer | 4K print buffer |

## General Considerations

An application may be placed at any CPU within the network. Consequently, application programs are portable to the extent they can operate in any CPU that is under control of an operating system in which the application was designed to be executed. These application programs need only to interface to the network facilities supported by the TPF System. Thus, a CPU which does not contain the TPF System must contain an interface program which adheres to the cross domain protocols supported by the TPF System. The Routing Control Parameter List (RCPL) when used as a message header, in essence, defines the TPF message routing protocol.

The term *System Control Program* (SCP) is loosely used throughout this document to indicate a control program other than the TPF system .

## Restrictions In the Use of Binary Synchronous Communications

The type of BSC lines used is determined by user requirements (existing networks, line tariffs, geographical locations, etc.) as modified by the following restrictions imposed by the TPF system.

The BSC protocol supported by the TPF system is as described in *General Information - Binary Synchronous Communications*, GA27-3004-2, with the following exceptions. TPF does not support:

- Start of Header (SOH) control character (except for Request for Test messages)
- Intermediate Block (ITB) control character (treated as a data character)
- Six-bit transcode transmission code
- Transparent USASCII transmission code
- Leading Graphics
- Switched lines
- Limited Conversational mode

The TPF system may be defined as a terminal to a *control point* through the use of lines controlled by the BSC protocol. In this case the TPF system acts as a point-to-point station, a multipoint control station or a multipoint tributary station. The line code on each line can be EBCDIC transparent, EBCDIC nontransparent or USASCII nontransparent. This allows the TPF system to be attached to leased lines or in-house BSC lines connected to a MVS system in a variety of network configurations. Several BSC lines can be used as a *link* between stations if desired. That is, a group of lines can be treated as an entity and the TPF system will queue output messages on the line with the smallest queue. Also, if an individual line goes down, all output messages will be rerouted to another line in the link.

## Local vs. Remote Routing

Message routing can be divided into two levels: Local Routing and Remote Routing.

Previously, local routing meant the situation in which an application is communicating with another application in the same TPF processor. This definition is expanded to include the routing of messages between an application residing in one processor of an loosely coupled complex and an application belonging to another processor within the same complex. For simplicity, the examples used to illustrate local routing in this document will show routing in a Single Processor System.

Remote routing comes into play when an application is communicating with an application or terminal outside of the loosely coupled complex. Remote routing, in this context, implies processor-to-processor communication where RCPLs are used in message headers to adhere to a format known as Processor to Processor Message Format. The processor-to-processor communications lines used for remote routing must be SDLC.

The distinction between local routing and remote routing is made on the basis of a CPU ID. Each control point in a TPF network must have a unique CPU ID. The CPU ID associated with a destination determines whether or not remote routing is used. Procedures are available for reaching a remote SNA domain without the use of PPMSG header. A PPMSG message header identifies the remote routing performed by the TPF *Message Router.*

BSC lines used for local routing have each station permanently attached to one application program which is given all the data received from the station. This application can, of course, reroute the message but as far as the TPF system is concerned, the line is used for local routing. A BSC line cannot be used for remote routing.

## Processing Description

In terms of sheer logic, the purpose of the Message Router is to deliver a train of information to a destination, as specified by control information (destination address) in message headers. The origins of this data are other CPUs connected to the TPF system through SDLC lines, and applications or system programs requesting transmission services through the ROUTC macro.

The following information introduces some of the system records and tables used by the message router. This is necessary to understand the message processing description that follows. A more detailed structural description of this data is provided later in this chapter.

**CI0CO** Routing Control Block

The RCB is a record permanently assigned to every terminal in the system which is referenced by a symbolic line number, interchange address, and terminal address. It contains a system area for use by TPF programs and a work area for the user. The system area contains information such as terminal description, routing information, etc.

**SN0CT** BSC Station Name Conversion Table

This table contains routing information for BSC stations which are treated as terminals rather than processors.

**RC0PL** Routing Control Parameter List

The RCPL is associated with each TPF input or output message. It provides information about the origin, destination and characteristics of the message. It is created by a component of the Message Router and used by system and application programs.

**RC0AT** Routing Control Application Table

The RCAT is used to control the routing of messages in the computer network. It is used by the Message Router to determine the destination of a message, as specified in the RCPL.

**WG0TA** Terminal Address Table

This table is used to find the ordinal number for the Routing Control Block associated with a given terminal. In addition, the WGTA contains the terminal status, terminal type, Application Name Table (ANT) index and indicators.

**AN0NT** Application Name Table

This table is an array of the names of all active and inactive applications defined as existing in the system. A reference to the RCAT is maintained for each application.

**RV1VT RV2VT**
Resource Vector Table

This table contains the definition, status, and ANT index for every SNA Network Addressable Unit (NAU) defined in the network. The Application Name Table (ANT) index is a reference to the application name for which the Network Addressable Unit is logged in. The RVT is used by TPF system programs to control the status and flow of messages through the system.

Within TPF books, an SNA Network Addressable Unit (NAU) is frequently misnamed a node. SNA NAUs have three attributes, that is, a name, a 16-bit network address and a type. The NAU type is used to identify network function and is either System Services Control Point (SSCP), Logical Unit (LU), Physical Unit (PU), or Line. SNA nodes are of four types, that is, CPU, communications controller, cluster controller or Terminal. The name of a node, using SNA formalities, is the name of the NAU of type PU assigned to an SNA node. The substitution of *Node* for NAU, although unfortunate, is usually clear in context. For example, the Node Name Conversion Routine should be called NAU Name Conversion since it converts the name of an NAU of any type to a 16-bit network address and vice-versa.

## Message Routing Example for Non-SNA Terminals

Multiple applications are active (capable of accepting and processing data) in the CPUs.

Since there is no activity in the system (no one is using the terminal), the system is idle and the communication control programs (CCP) is monitoring the lines for input.

System activity begins when a system user enters a message at a terminal. Since terminals can be used to access any application in the system, logically the first data input by the user would be a request to be connected to a specific application. Such a request is in the form of a log-in message specifying the desired application (for example, the application identified as APPL). As this message is received by the CCP in CPU A, the processing flows as summarized in Figure 7 on page 114.

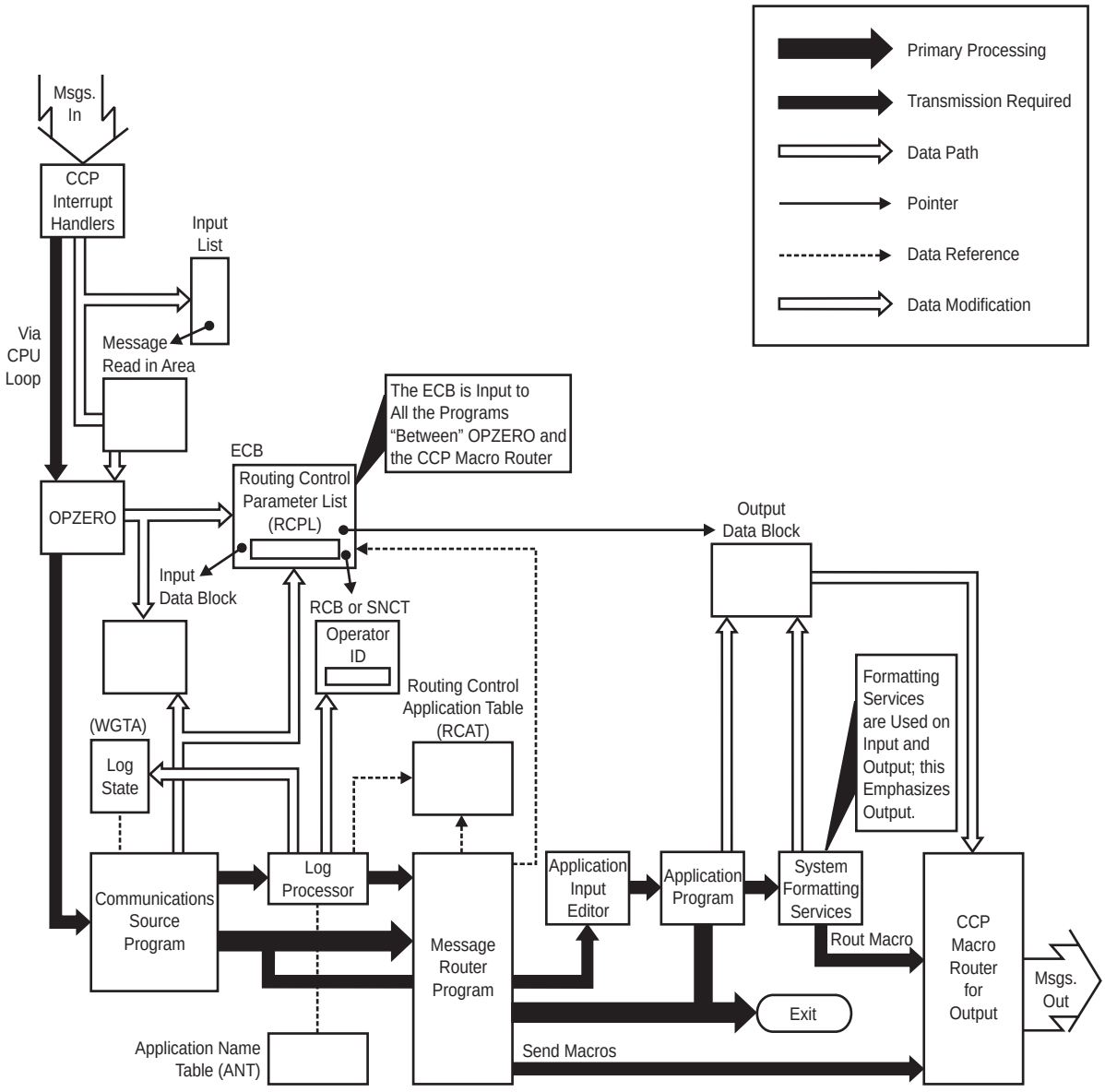



*Figure 7. Flow of a Non-SNA Message Through the TPF System*

1. The Operational Program Zero (OPZERO) creates an ECB, formats the data in an input message block, and attaches it to the ECB.
2. OPZERO gives control to the Communication Source Processor.
3. The communication source processor determines if the terminal is logged into an application by interrogating the WGTA slot associated with the inputting device.
4. Since the terminal was previously idle (not connected to any application), the destination of the data will be the Log Processor for path connection. The routing control block (RCB) is retrieved and a routing control parameter list (RCPL) is created showing the origin of the data as the terminal address and the destination as the Log Processor.
5. The Log Processor is given control. After validating the message, the WGTA is updated with the application name table (ANT) index of APPL, a response is generated, the origin and destination fields are reversed in the RCPL (response to the originator), and the *Router* program is given control.

6. The Router, after determining the terminal type to which the message is to be transmitted, invokes the appropriate TPF SEND processing routine for further data transmission processing.
7. A message stating a successful log-in to the requested application is received at the terminal.
8. Up to this point, we have emphasized the connection of a terminal to a specific application. All the processing has been confined to some components of the MR in CPU A. The terminal operator is now in a position to *converse* with the requested application, and he will do so by entering a message.
9. All the steps leading to the WGTA slot referencing by the Communication Source Processor are executed (Steps 1, 2, and 3).
10. The destination of this input message is determined by referencing the ANT index in the WGTA slot. This ANT index is actually a pointer to the application name (APPL) in the Application Name Table. The Communication Source Program references the Routing Control Application Table (RCAT) entry for APPL (pointed to by the ANT) to determine the location of the destination application APPL. An RCPL is created showing the origin of the data as the terminal address and the destination as APPL. The Routing Control Block (RCB) associated with the inputting terminal is retrieved also, if required.

    (The possibility of APPL being inactive is not taken into consideration here because if that were the case, the Log Processor would have rejected the previous request for terminal connection.)
11. The RCAT table would then provide the ENTER to the APPL Input Message Editor. (The Input Message Editor for an application is the program which understands the format and the significance of all the messages directed to the application.)
12. Ultimately, the APPL Input Message Editor calls (through an ENTER macro) an application program to accomplish the unique interpretation of the textual content of the message.
13. The application program, after generating an output message as a response to the terminal (possibly using the system formatting services) uses a ROUTC macro to transmit the output message. Normally, an application will avail itself of the system formatting services as specified in the Mapping Support Package. This package constructs, formats and transmits data streams on behalf of an application, thus rendering the application independent from physical device characteristics.
14. A parameter of the ROUTC macro is a pointer to the RCPL which would have been set up by the application showing the origin as APPL and the destination as the terminal address. The application would have used the simple technique of primarily reversing the origin and destination fields in the original RCPL created by the Communication Source Program and transferred to APPL.
15. As a result of the ROUTC macro being executed, a component of the Router is invoked which, after determining the type of the terminal specified as the destination in the RCPL, transfers control to the appropriate TPF SEND processing routine for data transmission.

The above described process continues message after message, transaction after transaction, until the terminal operator enters a *log* message indicating a request for:

a. Disconnection from APPL, in which case the Log Processor would update the terminal's WGTA entry to indicate an *idle* condition.

b. Connection to another application (for example, ABCD). This would be interpreted as request (l) plus a request for connection to application ABCD, which would be processed as in Steps 4 through 7.

## Message Routing from SNA Terminals

A log in message from an SNA terminal reaches a software package within the VTAM CMC called Unformatted System Services (USS). Through the facility of the Systems Initialization Package (SIP) and the Offline SNA Table Generation (OSTG) program, a remote TPF application, to be reached by an SNA terminal over an inter-processor link is identified as a local SNA resource of CPU A. Upon receiving a session initiation command, the TPF system binds the terminal to the LU within CPU A. For data (FM) messages that follow the session initiation commands, the Communications Source Program invokes the same logic described in the steps following 8 in the previous example. (This logic represents the LU with which the terminal is in session.)

The RVT, rather than WGTA, is used when the input is received from an SNA terminal. The router logic uses SOUTC macros for SDLC transmissions. Furthermore, address conversion (RID to pseudo LNIATA) is performed if the application indicates that it depends upon the use of an RCB.

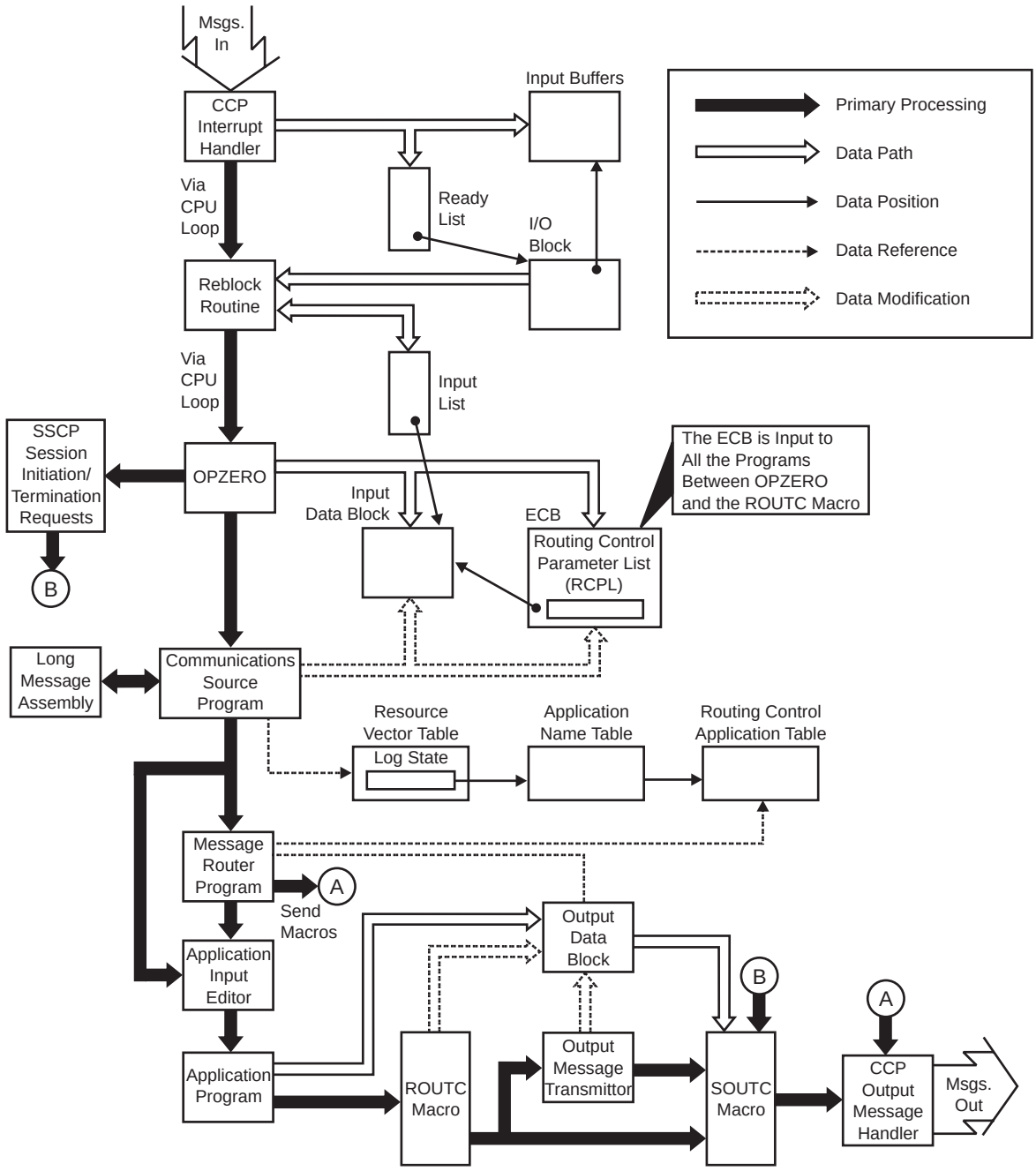



*Figure 8. Flow of a SNA Message Through TPF*

## Message Format

The basic type of message format used by the Message Router is AM0SG, which is the Application Message Format.

The AM0SG format is used when the TPF system passes a message to an application's Input Message Editor program for processing, and when an application program (or the System Formatting Services Mapping Support Package, on behalf of an application) passes a message to the TPF system for routing to its destination (either a terminal or another application).

The Communication Control Program (CCP) components that are processing data for various line disciplines handle message blocks in different formats depending on the line protocol and the technique used to process I/O operations on the communications lines.

As a result, various message block formats and definitions are used by the CCP components interfacing with the MR facilities. Figure 9 on page 119 describes the message format interfaces as they apply to the MR components overviewed prior to this section.

## Application Message Format (AM0SG)

The Application Message Format, together with the Routing Control Parameter List format (RC0PL), constitute the Application Program Data Interface to the TPF system.

When an application program receives a data message from the MR, the input message as described in AM0SG is attached to the ECB at data level zero and the RC0PL, which may be of variable length, is in the ECB work area as described in RC0PL.

When an application routes a message to a terminal or an application program, the AM0SG format is used as input to the ROUTC macro. The program interface is found in *TPF General Macros*, describing the ROUTC macro. The AM0SG format is also used by the System Formatting Services (Mapping Support Package), which formats and controls the output data for an application program.

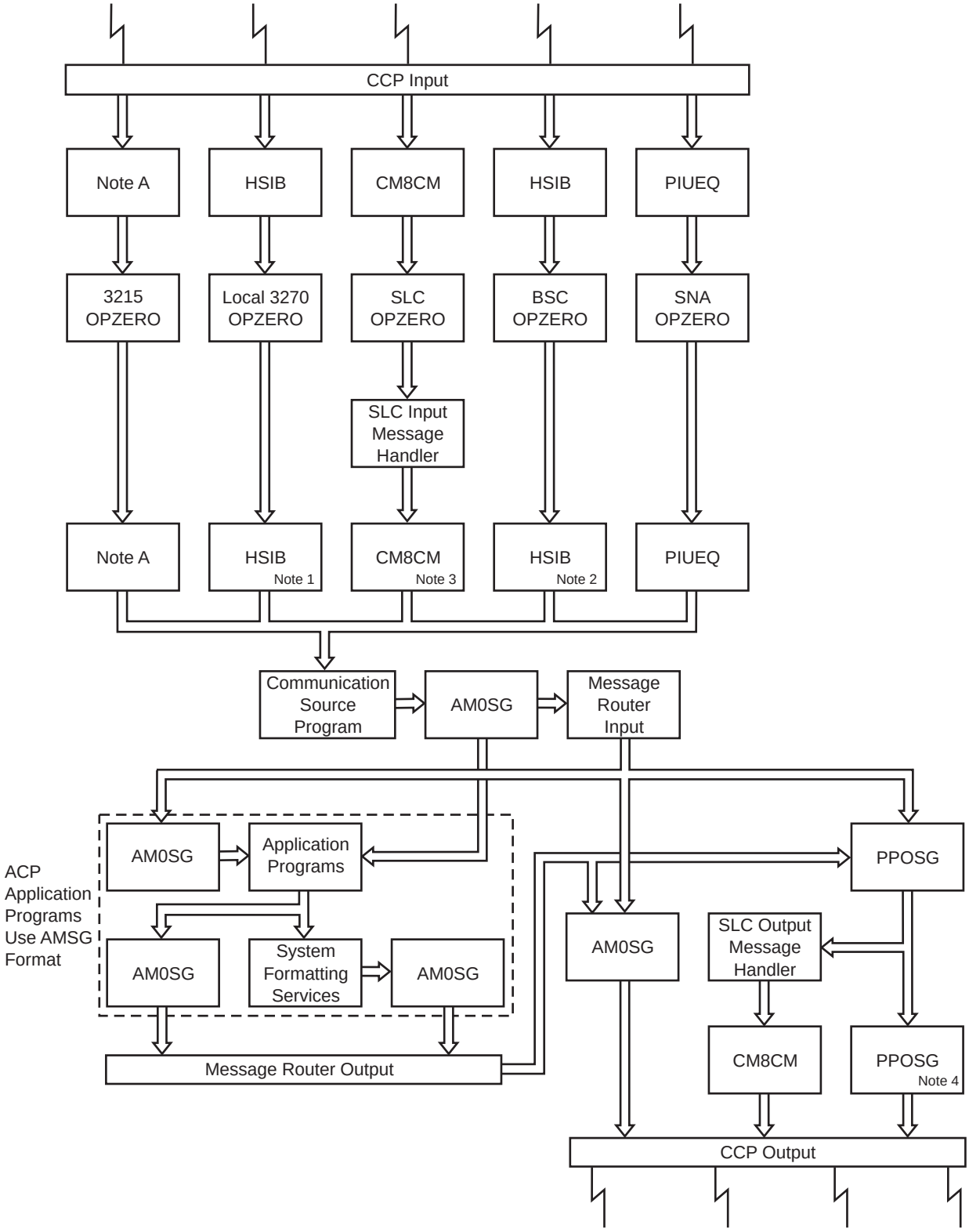



*Figure 9. MR Message Format Interfaces*

**Notes:**

1. There is no document describing the format of message blocks for the directly attached terminal operations (i.e. 3215/1052). Information relative to I/O programs can be found in 03-CCPNUC. The DSECT used is CM0MSG defined within the macro LINEQ.
2. The DSECT used is CM5MSG, defined within the macro LINEQ.
3. The DSECT used is CM0MSG, defined within the macro LINEQ.
4. The DSECT used is CM8CM. Each message begins with a header.
5. The DSECT used is CM5MSG, defined within the macro LINEQ. Each message begins with a header.

## Communications Source Program

All communication with the various sources of input to the TPF system is controlled by the Communication Control Program (CCP). The sources of input come from either a communication link to another CPU or from lines attached to remote concentrators. Various line disciplines are used (that is, NEF, SLC, BSC, SDLC) which imply unique differences in channel programs and procedures used by the CCP. For the purpose of focusing on the objective of this chapter, the CCP structure used in the execution of the many communication I/O requests will not be described here. All the details are given in the referenced TPF books. The component of the CCP which introduces all messages into the system and interfaces with the MR is the Operational Program Zero (OPZERO). The point of interface is the Communication Source Program. OPZERO is a collection of programs which are used to perform primarily the following functions:

1. Create and initialize an Entry Control Block (ECB).

   The initialization includes the CPU identification field from information contained in a system keypoint. The CPU ID is used to give uniqueness to system entries within the domain of one CPU. For example, RCBs are uniquely identified by terminal address and CPU ID since different terminals in the computer network may have identical addresses when controlled by different CPU's.
2. Link the message block to the ECB.
3. Execute an ENTER macro to invoke the Communications Source Program.

   The exception to this is the OPZERO program processing input from SLC lines which initially directs the data to either the SLC Input Message Handler or the SLC Link Controller. This is necessary to perform functions which are directly related to the line protocol. (The same logic is not needed for BSC lines since all the necessary control can be performed at interrupt time).

   Different entrances to the Communications Source Program are selected by different OPZERO programs depending on the source of input. The main objective of the Communications Source Program is to place input data messages into common system format and create an RCPL. The RCPL contains all the information needed to select an application editor for further processing or to have the Message Router facility select a network path for further message routing. The Routing Control Parameter List (RCPL) is placed in the work area of the ECB. While different entrances to the Communications Source Program make use of different paths within the program itself, the primary logic and associated functions can be described commonly. Figure 10 on page 121 represents an overview of the Communications Source Program.

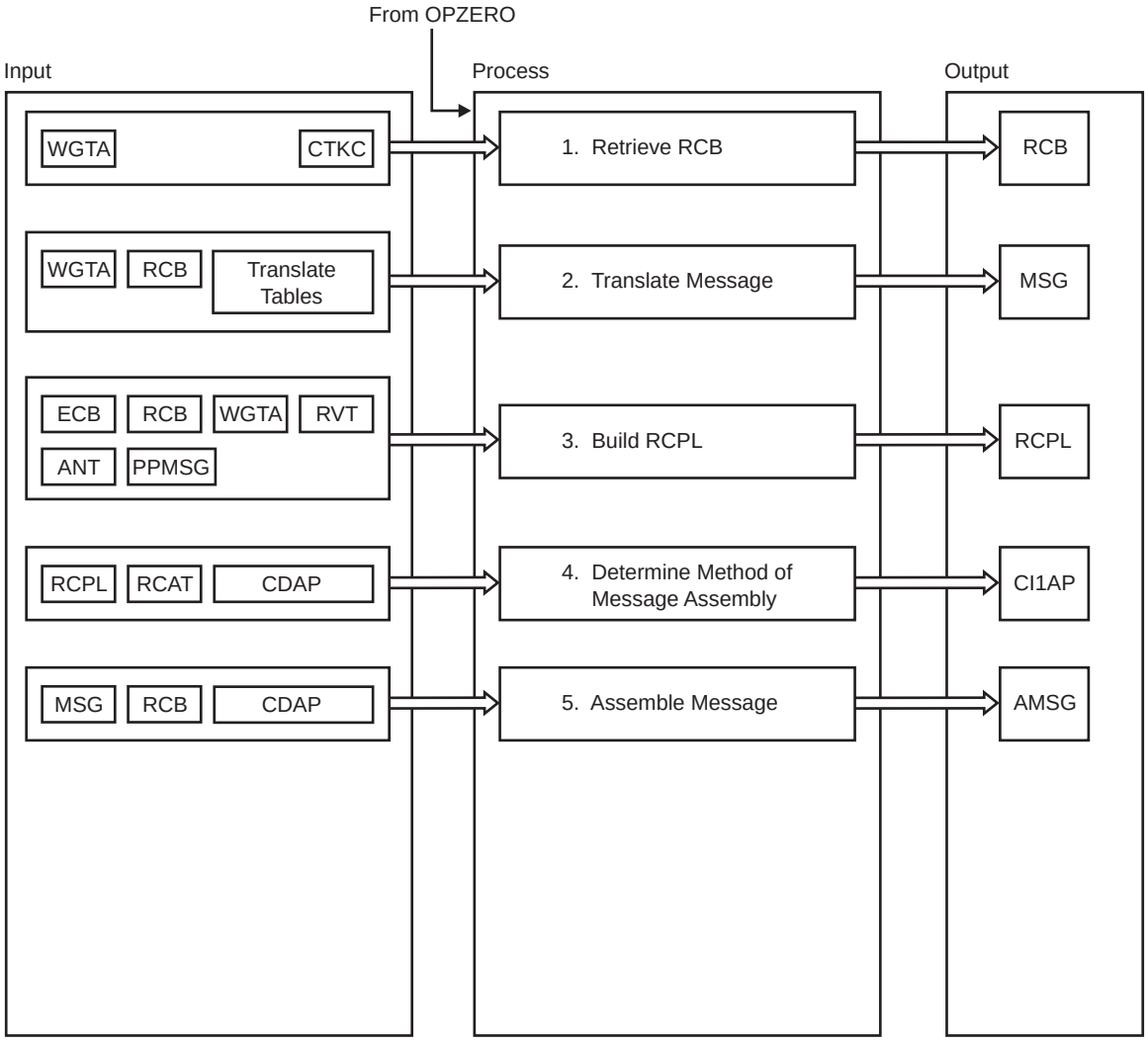



*Figure 10. Communications Source Program Overview (Part 1 of 2)*

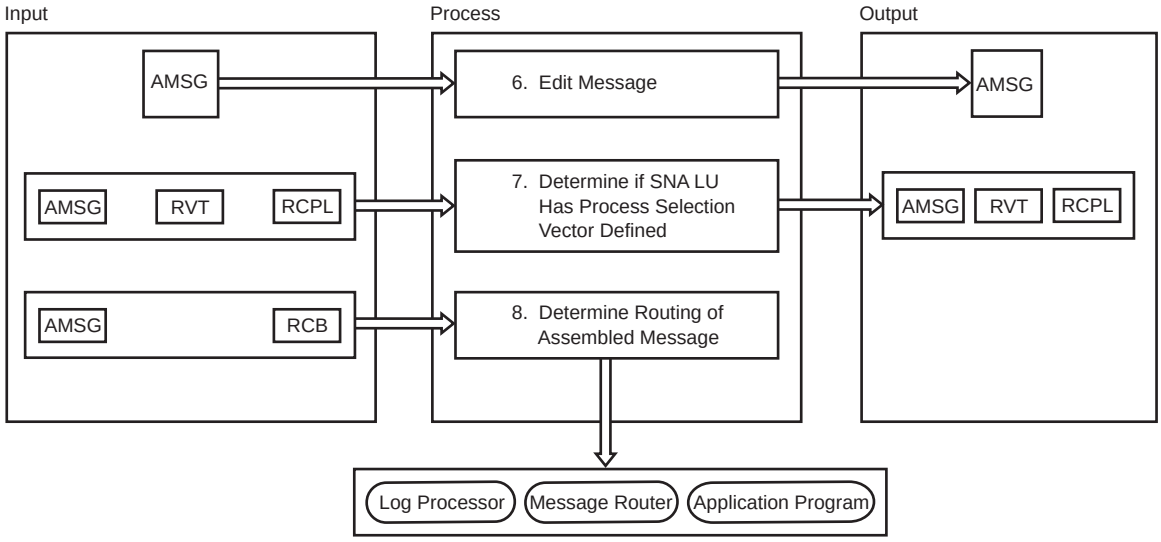



*Figure 10. Communications Source Program Overview (Part 2 of 2)*

## Processing Description of the Communications Source Program

**Step 1a:**

Locate the WGTA (Non-SNA Non-BSC) or the RVT (SNA) Slot.

Upon initial entry, the inputting terminal's WGTA slot is located through 03-WGR for non-SNA, non-BSC input. The WGTA contains pertinent information necessary to properly process and route the input request, such as terminal type, ANT index of the application to which the terminal is logged, and status indicators. SNA input will cause the RVT slot to be used, and like the WGTA, contains information required to process the input request.

**Step 1b:**

Retrieve The Routing Control Block (BSC)

As previously mentioned, an RCB is permanently assigned (in the fixed file area) to each NEF terminal in the system (computer network system). In addition, an RCB is assigned to each Symbolic Line (SLN) used in a BSC link for the purpose of maintaining common data relative to multiple transmissions such as message header information and data chaining fields. No RCBs are assigned to SLC links, BSC Stations or SNA Logical Units (SDLC terminals) with the single exception of 3270 SDLC terminals. The CPU ID renders the RCBs unique within the domain of one CPU. Therefore each RCB is identified by an address (LNIATA or SLN) and a CPU ID. In order to retrieve a RCB, the Communications Source Program invokes the Read RCB program (03- WGR). Using the address and CPUID of the given input source, the RCB ordinal number is obtained. 03-WGR obtains a file address from FACE (03-FACE) and retrieves the requested RCB. If the source of data is a BSC link, the link RCB may ultimately be substituted with a terminal RCB if, as described later, the origin is a NEF terminal and the destination is an application active in this CPU. The RCB for a BSC link is retrieved only if a multiple block transmission is received and is not passed to the application program.

**Step 2:**

Translate Message

The input data is translated to EBCDIC based upon the source of input. If the message comes from lines attached to remote concentrators (NEF or SLC lines) the WGTA provides information relative to the type of terminal and the translate tables needed to reduce the data to EBCDIC code.

Messages originating at 2915/4505/1977 terminals may contain backspace/erase characters in the data sequence. During translation these characters are identified and ultimately the Block Check Indicator (BCI) byte following End of Message is set to reflect this condition.

Messages originating from an SLC link are translated according to the technique used for SLC lines (see 18-SLC). At this point, SLC type B traffic (message switching) is assembled and ENTER macro executed which will give control to the message switching input message assembler (03-XIMA) for further processing.

Messages originating from a BSC link are translated if the BSC transmission code used is USASCII.

**Step 3:**

Build The Routing Control Parameter List (RCPL)

The RCPL is built in the work area of the Entry Control Block. If the source of input is a line attached to remote concentrators, information from both the ECB (Terminal address, CPU Identification) and the WGTA and ANT (Application ID, Terminal Type) is used to create the RCPL. If the source of input is a link connection (SLC, BSC) used for remote routing, the PPMSG header is used as a RCPL. In order to maintain continuity for functions supported in previous models of the TPF system, SLC lines may be used to link with systems not conforming to the MR standards such as SITA/HLN systems. If the source of data is this type of link, the RCPL is created using canned data generated by the user at system generation time). If the input source is a BSC line used for locally attached stations, the Station Name Conversion Table (SNCT) is used to build the RCPL. If the input came from an SDLC line, the Resource Vector Table is used to build the RCPL.

**Step 4:**

Determine Method Of Message Assembly

If message assembly is required, in order to eliminate excessive formatting of a message when its destination is not yet known, the Communications Source Program uses the Router Message Assembly Look-Ahead (03-RSAL) program to determine how an input message should be assembled. That is, what size main storage blocks and file records to use and what ID to use. The determination is based upon the message destination and the message type. For this purpose the RCPL previously created and the Routing Control Application Table (RCAT) are used to create a data assembly parameter list as defined by CDAP. This parameter list is used by the Communications Source Program to indicate to the General Purpose Data Assembly Program (03-CDAP) how to assemble a message (for example, chain in main storage or file, release input file addresses or not, block sizes to be used etc.).

**Step 5:**

Assemble Message

As previously mentioned, the General Purpose Data Assembly Program (03-CDAP) is used by the Communications Source Program to format and assemble input data before further processing is allowed to continue. This is done in order to place the message in a common system format (that is,

AMSG) and to perform assembly for the segments of a multiple segment message. Data in the process of being assembled is filed to secondary storage and the address of the filed block is recorded in the RCB, which is retrieved prior to activating CDAP. Input message processing terminates at this point (an EXIT macro is issued) if a complete message is not available to the Communications Source Program. Otherwise, the result of CDAP processing is an edited input message in AMSG format.

**Step 6:**

Edit Message

Messages from 2915/4505/1977 terminals may require text editing when so indicated by the check character. Editing consists of:

1. Removing NOP characters from messages originated at the RDI (Remote Display Interface) in the IBM 2946 Terminal Interchange.
2. Removing backspace/erase characters if present in the text.

As the editing is performed the message is restructured to account for the deletion of data.

**Step 7:**

Determine If SNA LU Has Process Selection Vector (PSV) Defined

If a Process Selection Vector (PSV) is defined for the originating SNA LU, the appropriate exit is activated. PSVs are used to convert from the various X.25 protocols to the TPF protocol and visa versa. PSVs may also be used for conversion to and from 3270 and 3600 protocols. For detailed information on PSVs, see *TPF ACF/SNA Data Communications Reference*.

**Step 8:**

Determine Routing Of Assembled Message

At this point a decision is made to invoke (through an ENTER macro) the next component of the MR facilities to further process the data. The next component is either the Log Processor, the destination application or the Message Router program. If the source of input is a BSC link connection, and the destination is not a locally resident application, Message Router program is invoked. If however, the destination is a locally resident application and requires a terminal control record (RCB), the BSC *link*. RCB is replaced by the terminal RCB, as described in step 1b. The application is then activated by executing a direct ENTER (defined in the RCAT).

If the source of input is a SLC link the same logic as for the BSC input is followed. However, when the SLC link is used to control indirect terminals (that is, SLC/HLN connection) the determination for routing the message follows the logic used for input originating at directly controlled terminals (lines attached to remote concentrators). Therefore, with the exception of indirect terminal support, input from BSC or SLC links is never processed by the Log Processor.

If the source of input is a line attached to remote concentrators (terminal input) then the decision to invoke the next component of the Message Routing facilities is based upon the following.

The Log Processor is invoked when:

- The terminal is in IDLE state (not logged in), or
- For Non-SNA terminals, the *sign-in-process* indicator is set in the WGTA and the first three characters of the input text are equal to *LOG*, requesting a specific LOG request.

All of the above conditions are recorded in the terminal's WGTA entry.

The message router program is invoked when the destination is not a local application (that is, remote application, local or remote terminal, or logical unit).

The destination application is invoked when the application is locally resident.

# Log Processor

The connection and disconnection of a non-SNA terminal to an application is performed by the Log Processor program. The terms connection and disconnection are loosely used to indicate the establishment of a data path from a terminal to and from an application. This is performed independently from the physical location of the application in the computer network. All data entered at the terminal for the duration of the connection is delivered to the application for processing.

Therefore, in order to converse with an application a terminal operator must first enter a *log-in* message. A *log-out* message is required to disconnect a terminal from an application.

There are two exceptions to the operator controlled process described above: 1) permanently logged terminals and 2) prime CRAS terminals.

Permanently logged terminals are terminals which are described to the system as having access to a specific application. LOGI/LOGO functions are restricted for these terminals. The Log Processor automatically logs these terminals *in* at restart time.

PRIME CRAS terminals are also permanently logged terminals. They are permanently logged into the SMP of their CPU. Unlike other permanently logged terminals, PRIME CRAS terminals have the ability to access other applications through the use of message prefixing. This function is activated by prefixing messages with the desired application name. For more details on this function see the System Message Processor (03-CSMP).

**Note:** SNA terminals requesting Session Initiation/Termination through log in (LOGON APPLID) and log out (LOGOFF) commands are handled on the SSCP session by the Unformatted System Services component of the VTAM CMC.

## Classification Of Non-SNA Applications

Applications in the TPF computer network are classified in one or more of the following categories:

1. Unrestricted

   This class of applications can be used by any terminal operator without restrictions.

2. Requiring *Sign In/Out* Procedure

   This class of applications require the user to follow specific *sign* procedures which are unique to the application itself. Therefore, a terminal operator must successfully *sign-in* to the requested application before the connection is completed and must successfully *sign-out* from the requested application before a disconnection becomes effective. Application *sign* processing is not part of the Log Processor. An entry point is provided so that the application can notify the Log Processor of successful *sign* procedures. The notification is in the form of an application to application message transmission. Before a sign-in can be accepted, the terminal must be logged-in to the application.

## Classification Of Non-SNA Terminals

Non-SNA terminals can be classified by the type of authorization processing they require. Classification is defined in the AAA/RCB Initialization Table (UA1UA) and kept in the WGTA, and will correspond to one of the following categories:

1. Permanently Logged - The terminal is logged in to an application at system restart time. The terminal operator is restricted from entering Log-in/out requests.
2. Application Authorization List- The terminal is allowed to Log-in/out to a specified list of applications. This list resides in the Terminal Application Authorization List (TAPP) and can be maintained through the use of the ZAUTH command.
3. Unrestricted - The terminal is allowed to Log-in/out of any application.

## Log Messages for Non-SNA terminals

The following are valid log messages.

---

**LOG IN: a. LOGI** *xxxx*

**Where:**

*xxxx*
A 4 character application name (the name of the application to which the terminal operator wants to be connected).

**Explanation:** None.

**Error Response:** None.

**System Action:** None.

---

**LOG OUT: b. LOGO**

**Explanation:** Used to Log out of the application currently logged to and set the terminal to IDLE state.

**Note:** A LOGI message can be used as a combination message to log out of the application the terminal is currently logged in to and log in to another application.

**Error Response:** None.

**System Action:** None.

---

**DISPLAY c. LOGU** *xxxx* **UNSOLICITED MESSAGE/S:**

**Where:**

*xxxx*
A 4 character application name (the name of the application to which the terminal operator wants to be connected).

**Explanation:** This message is used to display an unsolicited message originated by the application referenced in the message and directed to this terminal. More details on this type of message will be given in the section describing system usage.

**Error Response:** None.

**System Action:** None.

---

**PURGE d. LOGP** *xxxx* **UNSOLICITED MESSAGE/S:**

**Where:**

*xxxx*
A 4 character application name (the name of the application to which the terminal operator wants to be connected).

**Explanation:** This message is used to purge unsolicited messages originated by the application referenced in the message and directed to this terminal. More details on the use of this type of message will be given in the section describing system usage.

**Error Response:** None.

**System Action:** None.

## Processing Description of the Log Processor

The Log Processor is activated by the Communications Source program when one of the following conditions exists as specified by indicators in the WGTA entry associated with the terminal inputting the data:

- The terminal is in IDLE state (currently not connected to any application).
- The *sign-in-process* indicator is set in the WGTA and the first 3 characters of the input text are equal to LOG.

The following information introduces the concepts of screen formatting as it relates to the 3270-type terminals.

Display data, print data, or control data that is sent from the CPU to a 3270-type terminal is received and formatted at a control unit buffer and is then transferred to the device buffer. Conversely, display or print data that originates at a device is first sent to the control unit buffer, prepared for transmission, and then sent to the CPU.

A 3270 terminal can operate in one of two basic modes:

1. With an unformatted display in which no attribute character (and, therefore, no display field) has been defined. The terminal is used in a free-form manner similar to the IBM 4505.
2. With a formatted display in which a display field (or fields) has been defined as a result of storing at least one attribute character in the display buffer. The attribute characters define data fields with such characteristics as: protected (fixed format), unprotected (variable input data), alphameric input, selector pen detectable, and so on.

As a result of operating in one of the above modes, the input data sequence assumes different aspects. The presence of a set buffer address code (SBA) following the cursor address in the input data identifies the data sequence as being formatted. This knowledge is necessary to the Log Processor to be able to perform textual interpretation on the incoming data. (All the details relative to the 3270 terminal characteristics, functions and programming will be found in the referenced literature.)

In summary, the Log Processor has to be able to interpret data sequences input at non 3270-type terminals, and formatted or unformatted 3270-type terminals. Different components of the Log Processor analyze input data depending on the type and status of the terminal (these characteristics are reflected in the terminal's WGTA entry). This logic is schematically represented in Figure 11 on page 128.

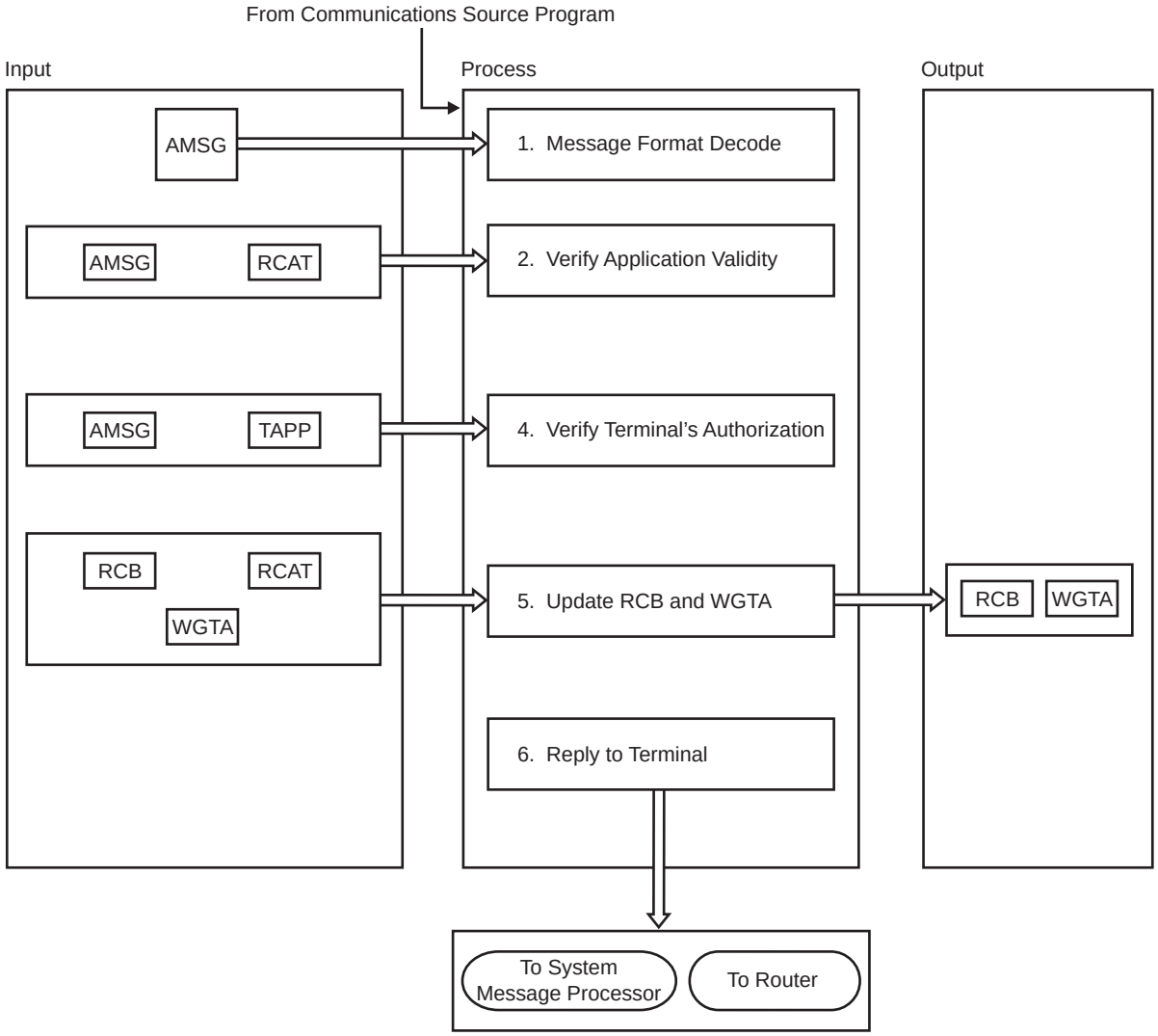



*Figure 11. Log Processor Overview (LOGI/LOGO Message Processing)*

**Information about Figure 11**

**Step 1:** The format of the message is identified by analyzing the type of terminal and, if a 3270-type, the mode of operation (4505 simulation or native mode). Invalid Log messages are discarded and the inputting terminal notified. All messages sent by the Log Processor are transmitted by activating a component of the System Message Processor (03-CSMP).

**Step 2:** The application name specified in the Log message is located in the RCAT table and its validity is verified. This is done by checking the application status.

**Step 3:** For terminals requiring authorization through the application list, the Terminal Application Authorization Record (TAPP) is retrieved and scanned. The application requested is verified by locating the supplied name in the terminal's Authorization List.

**Step 4:** The WGTA entry is updated to record the connection of the inputting terminal to the requested application.

**Step 5:** A response notifying the inputting terminal of a successful log is then generated and transmitted to the terminal.

LOGU and LOGP messages are processed as in Steps 1, 2, and 3 above before activating the Unsolicited Message Processor which will execute the requested function.

Some applications are classified as requiring *sign in/out* procedures. This requirement is recorded in the RCAT table at the entry relative to the specific application.

In summary, the most important function of the Log Processor is to store the requested application reference in the WGTA. This is key to the entire TPF-MR structure since this application name is used by the Communications Source Program to build the Routing Control Parameter List, which is in turn used by the Router program to find a path for the data from the terminal to the application.

## Message Router Program

The function of the Message Router program is to deliver data sequences to a destination as specified to it by control information (that is, the destination address). The origin of this data can be an external source (outside the logical boundary of the CPU) or an internal source (inside the logical boundary of the CPU). External sources are either lines attached to remote concentrators or communication links to other CPUs. Internal sources are either application or system programs requesting transmission services through the ROUTC macro.

The Message Router Program interface with external sources is either the Communication Source Program or the Log Processor. The Message Router Program interface with internal sources is the ROUTC macro interpretation routine as it will be explained below. Figure 12 is a representation of these interfaces.

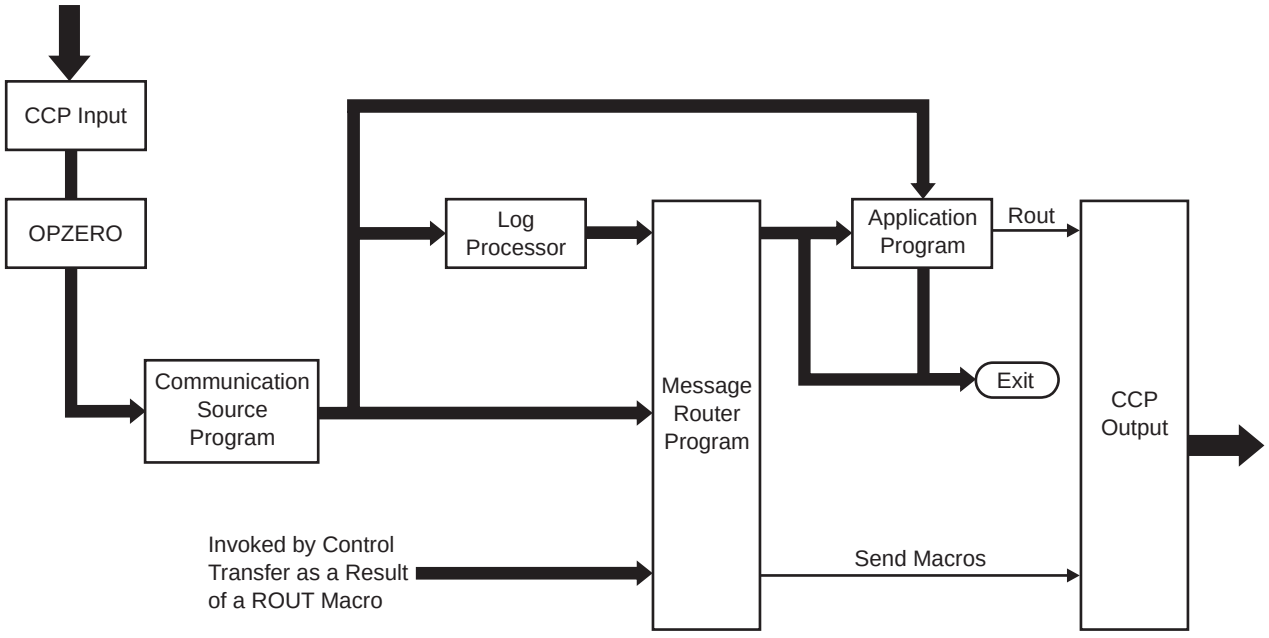



*Figure 12. Message Router Program Interfaces*

## The ROUTC Macro

The ROUTC macro is used by application and system programs to transmit a data sequence to a terminal or to an application. When the ROUTC macro is issued, the data to be transmitted is pointed to by a Core Block Reference Word in the ECB and the destination is included in a Routing Control Parameter List (RCPL) pointed to by a specified register. The data is in the format specified in AMSG, the Routing Control Parameter List is described in RCPL. Normally the RCPL would have been associated with the data input to the application's input message editor and passed to it by the Communications Source Program as we shall see. In this case the application program responding to the input would normally reverse the origin and destination fields in the RCPL before issuing the ROUTC macro. If a ROUTC is issued by a Subsystem to an application residing in a different subsystem of the same processor, the message is copied from the originator's file pool to that of the destination.

If a ROUTC is issued by a non-BSS Subsystem to a destination outside of the processor, it is first copied to the file pool of the BSS.

The macro interpretation routine will then either:

1. Invoke one of the TPF routines which interpret the SEND macros.
2. Issue a Control Transfer macro (CFXR) so that when the supplied ENTER expansion is executed at *post interrupt* time, a component of either the Message Router Program or the SNA Output Message Transmission Program is invoked for further processing. The RCPL would have been passed in the work area of the ECB and the data would have been attached to level 0 of the ECB.

Path (1) is chosen when the following conditions exist:

1. The destination, as specified in the RCPL, is a terminal, and
2. The terminal is not a 3284/3286-3, and
3. The CPUID of the destination is this CPU or a CPU within the same loosely coupled complex, and
4. The line address in the destination field of the RCPL is not an SLC pseudo line (indirectly controlled terminals) and
5. The line number is not 00 or 01, and
6. The data is a single segment message, and
7. The RCPL does not identify the data as an unsolicited or recoverable (SNA only) transmission.

When all of the above conditions are met, a SEND macro interpretation routine (SENDA, SENDB, SENDC, SENDL or SOUTC) is invoked depending on the terminal type.

The ROUTC macro, as previously described, can be used by an application program to transmit data sequences to another application. As shown in Figure 12 on page 129, the ROUTC macro interfaces directly with the message router program. Consequently, an application program is not required to log in to another application before it can transmit data to it. This is an important system wide general rule.

## Processing Description

As mentioned before, when the message router program is invoked (through processing of an ENTER macro) a standard interface is supplied. This interface is in the form of the data to be routed (in AMSG format) attached to level 0 of the ECB and the RCPL in the ECB's work area (beginning in the first byte).

For the purpose of simplifying the processing description, the message router program can be subdivided into two main components:

1. Local routing

   This component handles all routing requests when the CPU ID, as specified in the destination field of the RCPL, is part of the loosely coupled complex in which the Message Router Program is running.

2. Remote routing

   This component handles all routing requests when the destination CPU does not reside in the same loosely coupled complex as the CPU in which the Message Router Program is running.

The determination of which component to use is initially made by the Communications Source Program based solely upon the CPU ID of the destination.

**Local Routing**

In a loosely coupled complex, it is occasionally necessary to route messages between processors in the complex. These special cases require an additional step for local routing, the exporting of the ROUTC. That is, the ROUTC macro will be sent to the destination processor through the System Interprocessor Communication (SIPC) facility for processing. The processing logic then proceeds as represented in Figure 13 on page 133 and described below.

The type of destination, as specified in the RCPL, is analyzed and two different paths are taken depending on whether it is a terminal or an application.

If the destination is an application and the RCAT table indicates its availability, the application input message editor whose name appears in the RCAT entry relative to the specified application is invoked. The RCAT entry would contain the ENTER expansion for that editor.

If the destination is a terminal and the data being routed is an unsolicited transmission, as indicated in the RCPL, the unsolicited message processor is invoked for further processing unless the destination terminal is a receive only device.

If the destination terminal is a SNA logical unit and the message cannot be sent immediately (as in the case of a long message or recoverable message), the SNA output transmission program is invoked to send the message. For messages that can be sent immediately, the message is formatted and sent through the SOUTC macro.

If the destination terminal is attached through an SLC pseudo line (indirectly controlled terminal), the data is prepared for transmission through SENDK macro. This requires the use of additional control information (such as the High Level Network exit address (HEX), the remote link/system ID and the symbolic link number) needed by the SLC facility to perform and control data transmission. The message is then sent (through a SENDK macro) as an SLC high integrity message (a copy of the entire message is created on file).

For destination terminals attached through a NEF line the data is transmitted by issuing a SEND macro. The type of SEND (SENDA, SENDC, SENDL) used depends upon the destination terminal type as indicated in the RCPL. Multisegment messages destined for a

video terminal are transmitted by retrieving the message blocks from file and issuing multiple SENDCs.

**Note:** Message blocks are forward chained in the TPF file pool area.

If the destination terminal is a BSC station, its address may appear in the RCPL in one of two ways: as a symbolic address or as a symbolic name. If a four character symbolic name is used, it must be converted to a symbolic address before the SENDB macro can be issued. This is done by using the name to search the BSC Station Name Conversion Table (SNCT). If the name refers to more than one line (multiple BSC lines to the station), the line with the smallest queue is chosen and the SENDB macro is issued.

If the TPF system is unable to deliver a message to a BSC station, *line fallback*is invoked. This consists of using the destination in the RCPL to search the SNCT and determine if there is another line to the BSC station. The fallback process is automatic if a symbolic name was used in the RCPL. If a symbolic address was given instead, line fallback will be done only if so indicated by the terminal type field in the RCPL. If another line is found, SENDB is issued for that line. When all valid lines have been tried without success, the message is returned to the originator with the returned message indicator set in the RCPL.

The issuing of an ENTER (to an application) or a SEND (to a terminal) concludes the logical process of the local router component.

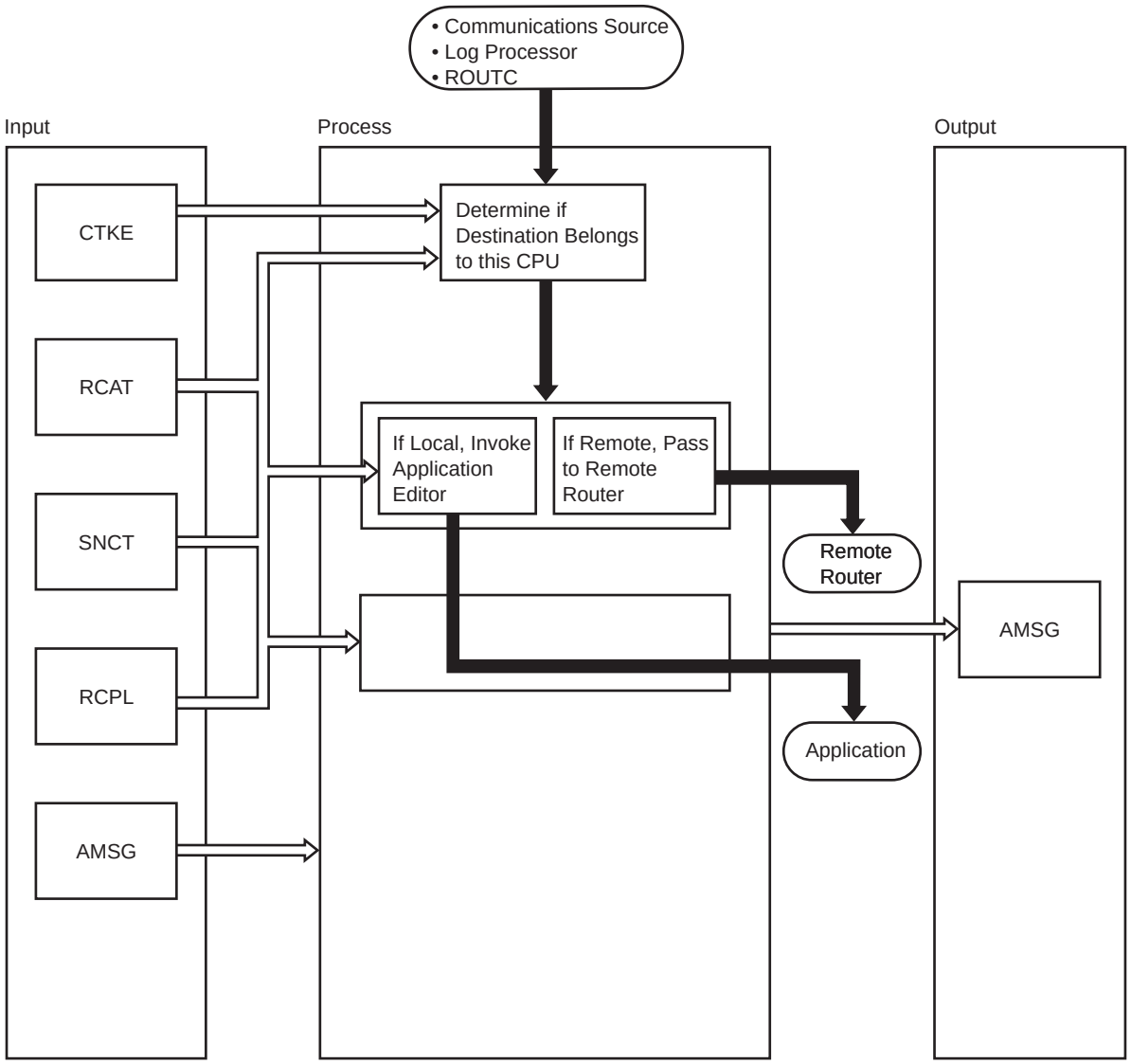



*Figure 13. Message Router Program Local Routing Overview*

**Remote routing**

When remote routing is required from a loosely coupled complex, all non-SNA traffic must pass through the EP processor. If the LC processor involved in the routing is a non-EP processor, the ROUTC is exported to the EP processor through the SIPC facility. Remote routing then proceeds as described below.

The SDLC path will be used when both of the following conditions are true:

1. Both the local and remote *domains* are the TPF system.
2. The Functional Management Message Router (FMMR) logical unit is defined and available in the remote domain.

If the SDLC path is not available, then the message will be returned to the originator.

## Remote Routing Through an SDLC Link

Processing of a message to be transmitted over an SDLC link to a remote domain involves the following steps:

1. Locate the Resource Vector Table entry for the remote Functional Management Message Router (FMMR) logical unit. Processing differs based on the type of destination:
   a. Destination is an application.

      The name of the application is given to the NAU Name Conversion program, which returns the Resource Identifier of the application. From here on, processing is the same as step (a).

   b. Destination is a terminal.

      The CPU-ID field of the RCPL is used to locate the FMMR entry for the CPU-ID.
2. Perform validity checks. The following checks are made:
   a. System must be in NORM state.
   b. The local and remote FMMRs must be in session.

      If any of these checks fails, the message will be returned to the originator.
3. Reformat the message. If the original message is in OMSG format, it is changed to AMSG format. If either the destination or the origin field of the RCPL indicates an SNA logical unit, the name of the logical unit is obtained.
4. Send the message. Messages transmitted over the SDLC link are always sent in the following manner:
   - The RCPL together with the general data area of the RCPL, if it exists, are transmitted as the *first-in-chain* PIU. If either the destination or origin field of the RCPL is an SNA logical unit, its eight character name is also transmitted in the *first-in-chain* PIU.
   - All user data blocks are transmitted as *middle-in-chain* or *last-in-chain* PIUs. This is to say that if the user sends chained 381 byte blocks, then each one of the 381 byte blocks will be transmitted as the *middle-in-chain* or *last-in-chain* PIU.

It should be noted that if the destination processor is a TPF system, ultimately the Local Router component in that processor will ENTER the requested application input message editor (see Figure 13 on page 133); that local router component may in fact be the Communications Source Program. The data in AMSG format and the RCPL are the standard input interface to the application.

Undeliverable Messages:

This component of the Message Router Program processes data that cannot be delivered to its destination by other components. Some of the reasons for not being able to deliver the data are:

- No available communications line/link toward destination CPU.
- Destination application name is invalid or application is not active.
- The data is for a terminal controlled by this CPU but the Message Router Program is unable to retrieve a file segment of the data due to I/O error.

The processing performed depends on the:

- Origin of the data.
- Destination of the data.
- Type of error encountered.

Application-To-Application Messages:

The returned message indicator in the RCPL is set. The origin and destination fields in the RCPL are reversed and the message is routed back to the originator. If the returned message indicator is already on, indicating that this message had already been returned and is on its way back to the originator, then no further attempt is made to route the message and the data is discarded.

Application-To-Terminal Messages:

If the error is encountered by the local router component (for example in the attempt to retrieve message segments from file) then a message is generated and sent to the destination terminal to inform the user of the *lost* message. If the error is encountered by the remote router component, Application to Application messages procedure is followed.

Terminal-To-Application Messages:

The terminal input message is discarded and a message is sent to the originating terminal informing the user of the inability to reach the application program. If the *terminal* is a BSC station, this is treated as a system error since the application must be active if the station is active.

The returned message indicator is a very important aspect of the Message Routing Facility because it influences the way application programs are designed. In fact the application input message editor has to distinguish between original data and returned messages and take appropriate action. More information is provided later in the chapter.

## System Services

Up to this point the emphasis has been on the architectural structure of the MR facilities, which is the main objective of this chapter. We have also introduced additional system facilities that are used to perform specific functions, such as the System Formatting Services, the System Message Processor (03-CSMP), and the Unsolicited Message Processor. While we will not indulge in any more detailed description of the System Formatting Services (also called *Mapping Support Package*), it is necessary to give additional consideration to the System Message Processor and the Unsolicited Message Processor because they interface directly with the MR facilities and have an impact on the system interface with the end user (terminal operators).

### System Message Processor

The System Message Processor Program (SMP) is viewed by the system and the system's users (terminal operators) as an *application*. The name of this *application* is SMP*x* where *x* is the CPU ID of the processor in which the program resides. This distinction is made to allow flexibility in the control of the computer network system as will be mentioned below. The implication of SMP being considered an *application* is that:

- An entry in the RCAT table must be initialized (during system generation) to contain the name SMP*x*.
- In order to have access to SMP, a terminal user must first log-in to it.

An exception to point (2) is made for the terminal designated as the Primary CRAS (PRC) for processor x. Whenever PRC requires access to SMPx, it is permanently logged into SMPx. A second exception to point (2) is made for applications inputting messages to SMP. This, as already explained, is a system wide general rule that applies to all application to application transmissions.

The primary function of SMP is to process operator commands ('Z' primary action code). Due to the nature of these messages, which are usually system control messages, only a selected group of terminals, identified in the CRAS table (CRAT), are allowed to input Z type messages. An exception to this general rule is made for a few messages. In TPF processor x, SMPx processes commands input to it and acts on system resources which are in the domain of processor x. In a computer network system, these commands may be received through a link from another processor. Consequently, a TPF processor y can be controlled by a terminal attached to a communication line controlled by processor x. This is possible if:

1. The RCAT in processor x has an entry relative to SMPy.
2. The terminal attached to processor x is authorized to use *application* SMPy. (Authorization verified through the log-in procedure.)
3. The CRAT table in processor y contains an entry for that terminal.

Another discrete function of SMP is to process system output messages.

In this role, SMP is regarded as the *output message editor* for all system programs. Eighteen (18) user entry points are the interface for message formatting and transmission. For example, the Log Processor uses SMP to output most of its messages.

## Unsolicited Message Processor

The unsolicited message processor (UMP) processes all the unsolicited messages addressed to a terminal. A message is classified as unsolicited when it is not generated as a response to an input message. The distinction between solicited and unsolicited messages is not made when the destination is an application rather than a terminal. Unsolicited messages originate at a terminal, in a TPF processor or in an SCP processor. They are identified as unsolicited by the textual contents of the input message if the origin is a terminal or by the information in the RCPL in addition to content if the origin is a program in a TPF or SCP processor.

The destination of an unsolicited message may be a specific terminal (single message) or a group of terminals (broadcast message).

For information about the unsolicited message processor, see "Unsolicited Message Processor" on page 37. The following describes how the unsolicited message processor interacts with the message router. Two major components of the unsolicited message processor process unsolicited messages in different stages:

1. Initial processing required to queue messages and send notification to the destination terminal.
2. Disposition request processing required to comply with the terminal user's request to either display or purge the unsolicited messages.

The initial processing component of UMP is invoked by the Message Router Program for processing unsolicited messages addressed to non-receive-only terminals directly controlled by this CPU. It is also invoked by SMP when an unsolicited message request is entered at a terminal. Since the request is in the form of a command, it would have been entered by a CRAS terminal logged in to SMP as previously described.

The Unsolicited Message Processor uses Unsolicited Message Directories (CODR) to maintain references to unsolicited messages until they are successfully transmitted or are purged as a result of a disposition request.

When processing an unsolicited message, a reference item is created and inserted in a directory record. The directory record is retrieved using the address of the destination terminal to compute the needed ordinal number within the directory's record type. A reference item contains information such as:

- Message and destination terminal characteristics
- Destination address
- Origin
- Message file address, etc.

Broadcast messages require the creation of reference items for the addressed terminals through the use of the WGTA table (WG0TA). Once the unsolicited message directory has been updated, a notification message is sent to the destination terminal to inform the operator of the existence of the unsolicited message. This notification (whose form depends on the terminal characteristics) is immediately sent for all broadcast messages. In order to schedule delivery of these messages, an Unsolicited Notification List (CONL) is created containing the addresses of each relevant receiving terminal. This list is processed by a component of UMP.

For non-broadcast messages (single messages) the destination terminal is immediately notified of the existence of the unsolicited message. After receiving notification, the terminal operator may either temporarily disregard the notice or immediately request a display of the message by following one of the procedures described in "System Usage Terminal Operations" on page 149. The appropriate input activates UMP which retrieves and searches the unsolicited message directory for the proper item based upon an originating application and the operator's terminal address. The unsolicited message is formatted appropriately and sent to the requesting terminal through the ROUTC macro, the Output Edit CRT Driver (03-UIO) or the Output Message Transmission Program.

A third functional component of UMP will, on a time initiated basis, police the Unsolicited Message Directory (CODR) to again advise terminal users of unsolicited messages queued for the terminal. This policing function will also purge, out of the system, any messages not displayed after several attempts at notifying the terminal user. The *purge* function may also be initiated by a direct request from the terminal user. It is described in "System Usage Terminal Operations" on page 149.

## Message Routing System Records

Many MR system records have been introduced in the discussion of the message routing facility architecture. Figure 14 on page 141 represents an overview of all the system records used by the message routing facility and their relationship to the various components. The records shown are exclusively used by MR. Other records that are common to other TPF function and have been mentioned in prior sections (for example, LSTB Line Status Table) are neither shown nor discussed any further in this book. Following is a summary of all the Message Routing system records containing information relative to data macro name, FACE ID, storage and file considerations, and record initialization. Additional detailed information is found in the referenced literature. Note that message router tables reside in the Basic Subsystem (BSS) only.

**Note:** Records that are listed here as *large* are 1055 bytes unless otherwise noted.

## CODR—Unsolicited Message Directory

**Data Macro Name:** C01DR

**FACE ID:** #C01RI

**Storage Factors:** Both prime and overflow CODR records are large records.

**File Requirements:** Prime records are allocated to fixed file storage. Overflow records are created and purged as necessary. They reside in random file pool storage.

**Record Initialization:** These records are initialized by the Unsolicited Record Initializer (DCSL-UMP) during the initial system cycle-up.

## CONL—Unsolicited Notification List

**Data Macro Name:** C03NL

**FACE ID:** Not Applicable

**Storage Factors:** The CONL records are large records.

**File Requirements:** These records are created and purged as necessary. They reside in random file pool storage.

**Record Initialization:** These records are initialized by the Unsolicited Message Processor (UMP).

## TAPP—Terminal Application Authorization List

**Data Macro Name:** TA0PP

**FACE ID:** #TAPID

**Storage Factors:** Both prime and overflow TAPP records are large records.

**File Requirements:** Prime and overflow records are allocated to fixed file storage.

**Record Initialization:** These records are initialized by the System Test Compiler (STC) or by the ZAUTH command.

## RCAT—Routing Control Application Table

**Data Macro Name:** RC0AT

**FACE ID:** #RCATU

**Storage Factors:** The RCAT record(s), when in main storage, reside in Global Area 3 of the Primary Subsystem User of the Basic Subsystem (BSS).

**File Requirements:** RCAT records are allocated to fixed file storage. They are forward chained, the last record having a forward chain value of zero.

**Record Initialization:** The records are initialized by Router Table Loader (03-RTLP) from the RCAT Initialization Table (RCIT) during the initial system IPL.

## RCB—Routing Control Block

**Data Macro Name:** CI0CO

**FACE ID:** #RCBRA

**Storage Factors:** RCBs are large records

**File Requirements:** The prime RCB's are allocated to fixed file storage. Overflow, if any, reside in random file pool storage.

**Record Initialization:** Each RCB is initialized, when its terminal/Link becomes active in the system, by the RCB Initialization Program (03-RCBI) from information contained in the AAA/RCB Initialization Table (UAT).

## RCIT—RCAT Initialization Table

**Data Macro Name:** RC1IT

**FACE ID:** Not Applicable

**Storage Factors:** The RCIT records are large records.

**File Requirements:** The RCIT records on file are large program records and reside in the program area.

**Record Initialization:** These records are initially generated by the System Initialization Package (SIP). Any changes to the application names will be reflected by changing and reassembling the RCIT records. This will force rebuilding of the Message Router Tables during the next IPL.

## UAT—AAA/RCB Initialization Table

**Data Macro Name:** UA1UA

**FACE ID:** #UATRI

**Storage Factors:** The UAT records are large records.

**File Requirements:** The UAT records are allocated to fixed file storage.

**Record Initialization:** These records are initialized by the System Test Compiler (STC).

## WGTA—WGTA Table

**Data Macro Name:** WG0TA

**FACE ID:** #WGTRU

**Storage Factors:** The WGTA table resides in main storage. The required area is carved out by the Control Program Initializer.

**File Requirements:** WGTA tables are allocated to fixed file storage. They are 4K records.

**Locking Requirements:** WGTA table entries must be locked for update as the entries are shared by I-streams.

**Record Initialization:** The WGTA tables are initialized by the WGTA Table Generation Routines (03-WGTA) during system cycle up.

## SNCT—BSC Station Name Conversion Table

**Data Macro Name:** SN0CT

**FACE ID:** Not Applicable

**Storage Factors:** The SNCT resides in main storage. It is created from the file records during system restart.

**File Requirements:** Up to eleven large program records on file are used to used to store the SNCT.

**Record Initialization:** The SNCT is generated by the System Initialization Package (SIP).

## RVT—Resource Vector Table

**Data Macro Name:** RV1VT

**FACE ID:** #RV1RU

**Storage Factors:** The RVT resides in main storage.

**File Requirements:** The Keypoint copy of the RVT resides in 4K fixed file records.

**Locking Requirements:** RVT table entries must be locked for update as the entries are shared by I-streams.

**Record Initialization:** These records are initialized by the SNA restart programs from the Resource Resolution Table, which is created by the Offline SNA Table Generation Program.

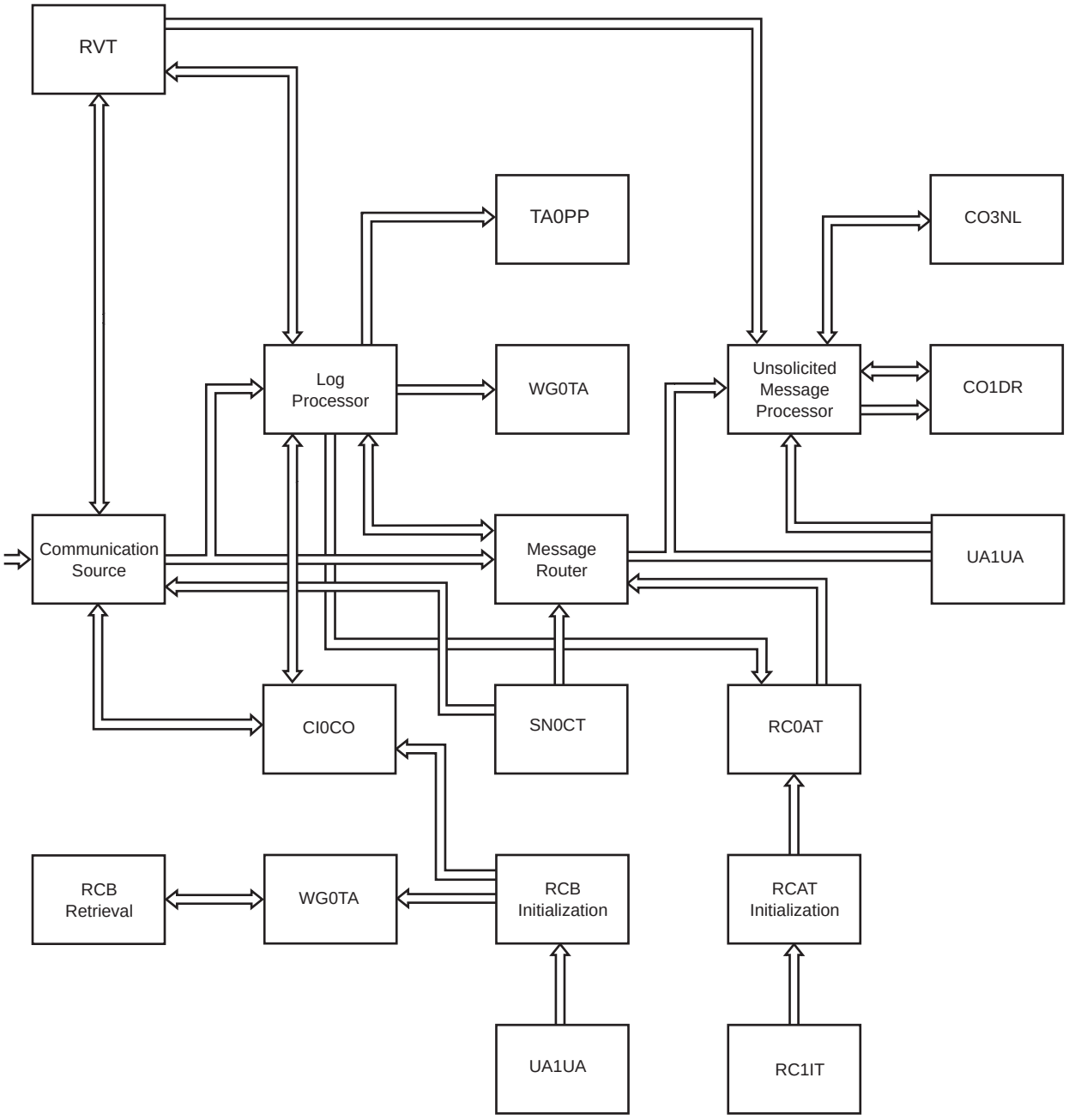



*Figure 14. Message Routing System Records. Relationship to MR Components.*

## Application Programming Considerations

The design of application programs to run under control of a TPF system with the message routing facility must take into consideration system interfaces which are different from those presented by prior models of the TPF system. These considerations, as it will be explained in this section, influence the way application programs are written. System compatibility with existing applications is maintained as outlined elsewhere in this chapter.

Three major areas of system interface can be isolated at the application level:

1. Input (to the application program).
2. Use of support facilities.
3. Output (from the application program).

This is described in *TPF ACF/SNA Data Communications Reference* for applications communicating with SNA logical units. However, the following is conceptually correct for all non-SNA terminals or SNA terminals utilizing a pseudo LNIATA address in a TPF system.

## Input Interface To Application Programs

As already described in prior topics of the chapter, the message router or communication source program ultimately invokes an application input message editor as specified in the RCAT. When the application input message editor is given control for the textual interpretation of the data, the following standard interface is presented:

- The Entry Control Block work area (beginning at EBW000) contains the Routing Control Parameter List as specified in document RCPL.
- Core Block Reference Word zero (CE1CR0) in the ECB contains the pointer to input data in storage.
- If required by the application, Core Block Reference Word three (CE1CR3) contains the pointer to the terminal's Routing Control Block in main storage if the input data originated at a NEF terminal. The Routing Control Block record is in HOLD status.
- If the data originated at a terminal, field CE1CPD in the ECB contains the ID of the CPU in direct control of the terminal. If the source of input is an application, field CE1CPD contains the ID of its own CPU.

  **The RCPL:** The content of the RCPL as input to the Application Message Editor contains useful information such as:

  – The address (LNIATA CPUID) of the terminal inputting the data, if the origin is a terminal. In prior releases of the TPF system the address (LNIATA) was found in field EBROUT of the ECB.

  – The type of origin (terminal or application).

  – The returned message indicator. This is a very important indicator and should always be checked by the Input Message Editor before any further processing is performed. It identifies data that is being returned to the originating application due to the system being unable to transmit the data as requested by the application through a ROUTC macro. Obviously, returned messages are processed differently from original data.

  – Terminal type identification, if the origin of the data is a terminal. This indicator can be used to decide the manner in which the input data must be processed if its format is dependent on the characteristics of the inputting terminal. An example would be if system formatting services are required to perform data mapping.

It should be noted that the RCPL should be preserved by the application for it is needed in order to transmit data (that is, reply to the input message). Since the system interface for data transmission is the ROUTC macro, it is not necessary to

keep the RCPL in the work area of the ECB. In fact, a parameter of the ROUTC macro identifies the name of the system register containing a pointer to the storage location where the RCPL is to be found.

**Input Data:** The standard format of the input data, as presented to the application input message editor is described in data macro AM0SG. Messages in the AMSG format are fully assembled and the pointer to the prime or only input block in storage is found in field CE1CR0 of the ECB. Overflow blocks are located in the random file pool storage area (short term) and forward chained to the prime block. It is the application's responsibility to manage the file pool storage area containing overflow AMSG blocks.

**The RCB:** If the data input to the application message editor originated at a NEF terminal, then field CE1CR3 in the ECB is used to point to the storage location of the terminal's RCB if required by the application. The RCB record is in HOLD status. A detailed description is found in data macro CI0CO. Note that the RCB is divided in two sections: a system section and an application section. The latter can be defined and used by applications as a terminal assembly area to maintain continuity or transfer information across multiple messages input from a terminal. Under no circumstance should the application store information in the system area of the RCB. This could lead to unpredictable results.

## Use Of Support Facilities

An application program can communicate with a display terminal using one of two basic methods. In one method, the display screen is left unformatted and is used in a free-form manner. In the second method, the display image is formatted by the application program by assigning control and data characters to a display image to be presented to the terminal operator. Printouts can be formatted in the same manner as a display image.

Of all the terminals supported in the TPF system, only the 3270 terminals can be used in both formatted and unformatted mode. In all other terminals, the images are left unformatted (such as the 2915 display).

A TPF system facility called Mapping Support Package can be used by application programs to construct and format application to application, application to terminal and terminal to application data sequences. As a formatter vehicle, this facility presents to the user program data which is devoid of device dependent code. This formatting capability is attained by user defined formats, or maps, which characterize input and output for display terminals and output for printer devices.

When used with input messages originating at field oriented terminals (for example, 3277 display station) the Mapping Support facility deletes device dependent control characters from the input data sequence and arranges the data in a symbolically addressable manner for the requesting user program. Similarly, output data sequences created by application programs are reformatted with the appropriate device control characters required to display the information on field oriented terminals (for example, 3270 displays) or to print the formatted data on hard copy devices (1980-24, 3284, 3286, 3287, 3288, 3289, 1977). More information on the services and use of the Mapping Support Package is found in the referenced books.

## Output Interface From Application Programs

All application programs running under control of a TPF system with the Message Routing facility have one single interface for transmitting data. All data is transmitted by application programs through the ROUTC macro. If the Mapping Support Package is used by an application to format output data sequences, this support package is capable of assuming the responsibility for issuing the ROUTC macro in behalf of the application itself.

**The ROUTC Macro:**

The detailed description of the ROUTC macro is found in the *TPF General Macros*. When this macro is issued by an application program (or by the Mapping Support Package on its behalf), the data to be transmitted must be in AMSG format and the routing information must be contained in the RCPL pointed to by the contents of a specified system register. If the data to be transmitted is the response to an input message, the original RCPL passed to the application input message editor by the Message Router program can be used as the RCPL for the output data by simply reversing the origin and destination fields and the origin and destination type indicators. If the data to be transmitted is not the result of processing an input message, the RCPL must be constructed by the application program in the format specified in data macro RC0PL. In this case, the message type indicator in the RCPL must indicate an unsolicited message if the data is addressed to a terminal.

## Sign/Log

It is important to clearly understand the difference between the *Sign* and *Log* functions in a system supported by the Message Routing facility of the TPF system. As already explained, the Log function is a system function and it is used by a terminal user to establish a data path from the terminal to and from an application. Terminal authorization requires the clearance of the terminal through verification of authorization. This clearance is performed, by the Log Processor, at the system level without the intervention of the application program. In other words, the establishment of the data path is conditioned to the terminal's authorization to input data to the application (which is different from being authorized to use the application).

The Sign function is an application function and may be required from a terminal user, connected through a *log* procedure, in order to use the application's functions. This may be needed for accounting purposes or for applications performing multiple functions each requiring different security codes or for a variety of other reasons. The class of applications requiring sign in/out procedures has a special interface which enables them to notify the Log Processor when a successful *sign* has been accomplished. This is necessary so that the data flowing from a terminal to the application between *sign-in* and *sign-out*. notifications does not need to be monitored by the Communication Source Program for a *Log* request. This special interface, however, applies only to applications receiving messages from non-SNA terminals and SNA terminals addressed by their 3270 pseudo LNIATAs. The interface to the Log Processor is identical for *sign-in* and *sign-out* notifications and is in the form of an application to application data transmission where:

- The origin of the data is the application itself.

- The destination of the data is application CLG*x* (*x* is the ID of the CPU in direct control of the terminal). If the application sending the notification resides in an SCP system, the CPU ID would have been passed to it as part of the origin of the data in the PPMSG header. If the application sending the notification resides in a TPF system, the CPU ID would have been part of the origin in the RCPL.
- The text of the data is the address (LNIATA CPUID) of the originating terminal.

A direct result of the *log* and *sign* procedures is that the Communication Source Program monitors all input to applications which do not require sign in/out procedures for *Log* requests. Consequently, no data message beginning with the characters *LOG* is allowed. As seen above, this restriction does not apply to applications requiring *sign* procedures.

## Message Routing Facility Installation Overview

The Message Router facility is always generated by SIP. This means that the *generation* of system records as well as the assembly of selected system programs which are part of the MR Facility will always be included at system generation time.

The system records used by the Message Routing Facility have already been discussed. Note that SIP generates only the following records:

1. RC1IT (Program records COHA, COHB, and so on)

   The RCAT Initialization Table contains user application entries and system *pseudo application* entries. The names of these system *pseudo application* entries are fixed (not optional) and are:

   - LOGI
   - SMP*x*
   - CLG*x*

   Where *x* represents the CPUIDs of all the TPF systems in the computer network. Obviously, these names cannot be used to identify other user applications.
2. SN0CT (Program record CRS1, CRS2,...)

   BSC Station Name Conversion Table.

The remaining records must be generated using the Pilot Generation function of the System Test Compiler (STC). STC language is used to generate the following record:

1. UA1UA

The WGTA Tables are initialized during system cycle up and do not require any offline generation.

All the details relative to the use of SIP and STC are found in *TPF System Generation* and *TPF Program Development Support Reference* respectively. Details relative to the type and value of the contents of the records to be generated are found in the specific record description documents and in *TPF System Generation*.

Figure 15 on page 146 represents a general overview of the process required to ultimately be able to IPL a TPF system with the Message Routing Facility. Note that the process shown is by no means a true representation of the steps and program components involved in the generation and loading of data and programs to the TPF on-line files. It only serves the purpose of pictorially drawing the reader's

attention to major areas of the TPF system which are directly involved in the installation of the MR Facility.

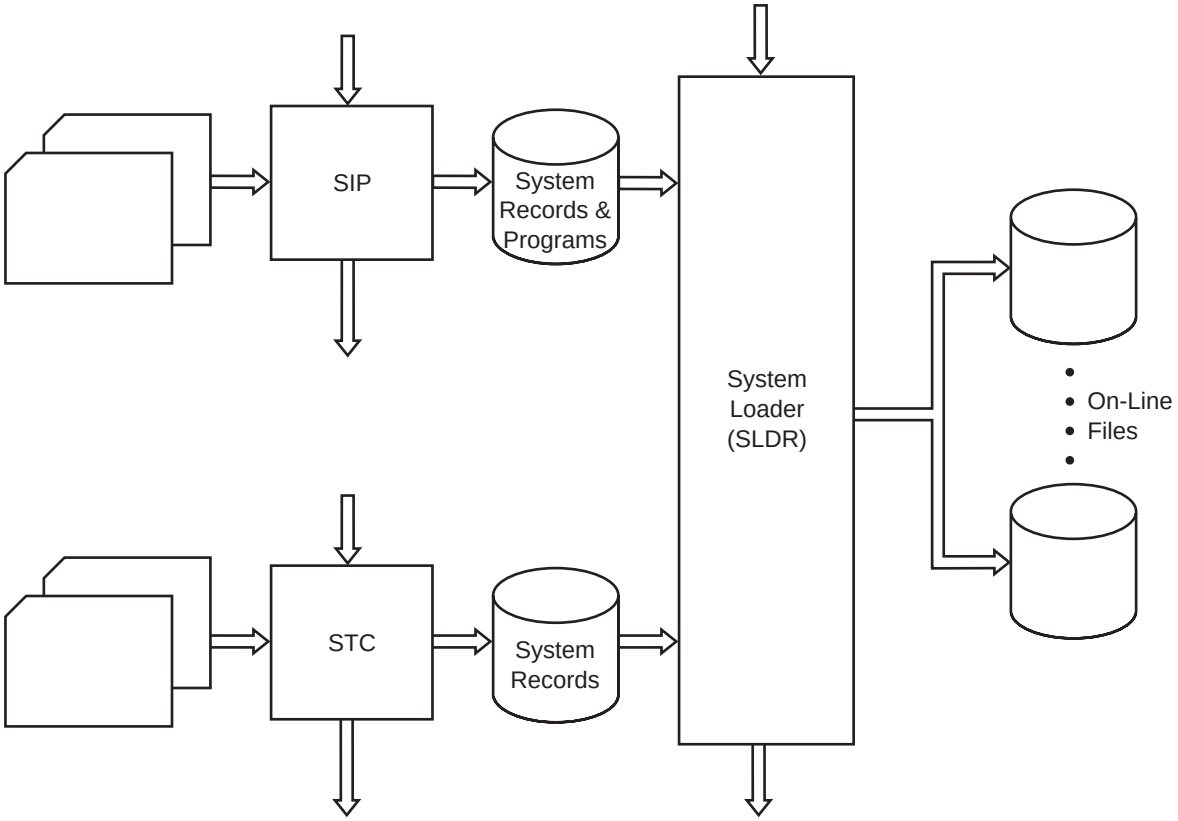



*Figure 15. AMR Generation/Installation Overview*

## System Operation

TPF system operations are generally independent from the Message Routing Facility. This is to say that the MR Facility does not alter system operations procedures which are outside its own domain. For example, procedures relative to a TPF system remain unaltered whether the CPU is a stand-alone processor or part of a computer network system.

The information in *TPF Operations* about the message routing commands describes, in detail, all of the facilities and the control language (commands) necessary to manage them.

This section is intended to review some of the operational procedures directly related to the MR Facility. It is assumed that all the phases associated with system generation and loading have been executed resulting in all the MR programs and system records to be stored in the appropriate areas on the online files.

## Initial IPL

The Routing Control Application Table (RCAT) records are built as a result of the initial (first time) IPL of the system, from the RCAT Initialization Table (RCIT) records. Subsequent IPLs of the system will use the keypointed records from file.

Following the initial system IPL, the operator is notified of the completion of the system initialization by the advisory message `hh.mm.ss. RESTART-1052 STATE`. At this point the operator would initialize the Routing Control Blocks by entering the appropriate command. Note that RCB initialization is only required after the initial system IPL. The need for selective RCB initialization may arise in the course of normal system operations in the event the system areas of an RCB is corrupted or as an emergency procedure if all the terminal log out attempts have failed.

## Activating the System

The operating procedures to *cycle up* the system and activate the various types of communication lines are independent of the message routing facility and are described in detail in *TPF Operations*. Procedures relative to the activation of SCP systems are to be found in the appropriate books. It is assumed, for the purpose of continuing the review of the procedures, that all the systems in the computer network have been activated and are operational.

## Starting an Application

An application whose name is in an RCAT entry can be started in any system state through a command of the type:

---

**ZROUT STRT** *xxxx*

**Where:**

*xxxx*
The name of the application.

**Explanation:** Obviously, before the application is started all the necessary operative procedures to make all the needed resources (for example, files) available to the application, must have been completed.

After the Start request has been processed, terminals on the network are capable of having *log-on* requests (to that application) honored by the TPF system.

**System Action:** None.

**Error Response:** None.

## Stopping an Application

An active application can be stopped in any system state through the use of the following message:

---

**ZROUT STPI** *xxxx*

**Where:**

*xxxx*
The name of the application.

**Explanation:** If the TPF system is in 1052 state the application status in the RCAT is set to STOPPED. When terminals are logged in, and the system is above 1052 state, a broadcast message is sent to notify the operators that the application is being stopped. The operators then have 15 minutes to finish what they are doing. When finished, the operators should log off. At the end of 15 minutes, terminals that are still logged in are automatically logged off and the application status is set to STOPPED. Permanently logged terminals are not affected.

**System Action:** None.

**Error Response:** None.

## System Restart

All the MR system records that during normal system operation reside in storage are keypointed through the Keypointing mechanism with the exception of the RPST and RCLT records residing in a non-EP processor in an loosely coupled complex.

Therefore, the MR restart programs in the restart schedule are capable of restoring the MR facility to the status in which it was prior to the restart occurring.

No special operational procedures are needed to restart the MR facility. Exceptional conditions may require intervention such as RCB initialization, stopping an application etc.

## System Status And Message Routing

At any instant, the processing capabilities of a TPF CPU depend on the system state (as defined in TPF literature) in which it is operating. When MDBF is included, the term *system state* refers to the state of the subsystem in which the application resides. In order to render the total computer network system relatively independent from the combined state of each CPU, the action taken by the Message Routing Facility on the data received from either internal or external sources, depends exclusively on the processing capabilities of its own CPU.

While this does not directly affect system operations, it is nevertheless opportune to be aware of these actions for they are dependent on whatever steps the system operator takes in any CPU relative to the changing of system states.

The MR action (reject or route) is shown in Table 7. In general, the following rules are observed:

- In 1052 state, only messages for resident System Message Processor (SMP) like applications will be routed. (SMP-like applications are user defined applications which have the characteristics of the TPF-provided *SMPx* program.)
- In CRAS state, messages for other CPUs are routed. Messages for the host CPU will be routed if they originate at either a CRAS terminal or from a program. (Host means any CPU in which the Message Router Program is in control).
- Above CRAS State all messages are routed.
- Application programs must provide their own checks if they have any System State Restrictions.

*Table 7. MR Action*

| **Message Destination** | **Message Origin Type** | **1052/UTIL System State** | **CRAS System State** | **MESW/NORM System State** |
|---|---|---|---|---|
| Terminal or Program in Another CPU | Not Applicable | Reject | Route (1) | Route (1) |
| SMP-Like Applications in the Host CPU | Program | Route | Route | Route |
| SMP-Like Applications in the Host CPU | Terminal (2) | Route | Route | Route |
| Application Not SMP-Like | Program | Reject | Route | Route |
| Application Not SMP-Like | Terminal (2) | Reject | Route (3) | Route |
| Application not SMP-Like in the Host CPU | Terminal (2) | Reject | Reject (3) | Route |
| Terminal Connected to the Host CPU | Program | Not Applicable | Route | Route |

**Notes:**

1. The SNA Functional Management Message Router (FMMR) is used for message routing, these requests will be rejected unless the system is in NORM state
2. CSMP will verify that the inputting terminal is in the CRAT Table.
3. Message Router will verify that the inputting terminal is in the CRAT Table.

## System Usage Terminal Operations

This section describes the language used and the actions taken by a terminal operator in order to gain access to the environment (application programs and system facilities) controlled by the Message Routing Facility.

## Log In

Input from any terminal not permanently logged in and not classified as the PRIME CRAS (PRC):

---

**LOGI** *xxxx*

**Where:**

*xxxx*
The name of the application.

**Explanation:** The terminal operator requests to be connected to the application referenced in the message.

**Normal Response:** LOGI COMPLETE

The connection has been accomplished.

**Error Response:**

1. INVALID FORMAT
   The format of the input message is in error.
2. INVALID APPLICATION
   The name of the requested application does not appear in the RCAT.
3. APPLICATION NOT AVAILABLE
   The application is currently not active (stopped).

## Log Out

---

**LOGO**

**Explanation:** The terminal operator requests to be disconnected from the application currently logged to.

**Normal Response:** LOGO COMPLETE

The terminal has been disconnected and it is idle. A log-in message may now be entered.

**Error Response:** NOT LOGGED TO APPL

The terminal is not currently logged in.

## Combination LOGO/LOGI Messages

A message of the type defined *LOG IN* can be entered from a non-SNA terminal to concurrently log off and log in to an application. Normal and error responses, as specified for the above messages, apply to the combination messages.

## Inputting Operator Commands

In order to input a command from a terminal, the terminal user must first log in to the System Message Processor. The application name specified in the log-in

message must be SMP*x*, where *x* is the CPU ID of the TPF system to which the terminal operator intends to send the commands.

As already described, the PRC terminal is always logged in to its own SMP.

## Send/Receive an Unsolicited Single/Broadcast Message

Terminal actions required to send or receive an unsolicited message are described in DCSR-UMP.

## Compatibility with Other TPF Releases

In order to maintain functional and operational compatibility with prior TPF releases, special considerations have been given to the use of existing applications (for example, airlines reservation) from terminals such a the 2915, 4505 and 1977. For this purpose, all application programs whose input message editor is UII (03-UII) are indicated in each RCAT as a RES-type application. When an agent on a terminal wants to access a RES-type application, he must first request log-in to the application, followed by a *sign message*. Because of the special considerations given to the RES-type applications, many MR components contain certain code used to process input from terminals connected to RES-type applications. For example, the Agent Assembly Area (AAA) record is retrieved by WGR rather than the Routing Control Block (RCB), messages are input to RES-type applications in IMG format (IMG), no RCPL is initialized, and so on.

# Output Message Transmitter and Input Message Editor

The user interface (UI) package allows communications between 1977, 2915, 4505, and 3270 terminal operators and application programs in a reservation environment. The interface consists of editing the input message for presentation to the application and ensuring that output messages from the applications are correctly edited and formatted for the specific terminal type.

## Functions

- To edit all input and output messages associated with the applications environment.
- To process move scroll requests (M action codes) from terminal operators.
- To process repeat last message (R action codes) from terminal operators.

## Component Programs

## Programs In This Package

**Input Message Editor**

This program distributes the various input messages to the appropriate applications driver.

**Output Message Edit CRT Driver Program**

This program does a basic edit of all output messages submitted to the package by the applications programs to determine:

1. Whether the Agent's Assembly Area (WA0AA) and/or the Routing Control Block (CI0CO) is in core.
2. Whether a canned message is required.
3. Whether the response is to be sent to a hardcopy terminal or a CRT. The program also adjusts the output message format for transmission to a CRT terminal.

**Scroll Control Program**

This program maps, by line and OMSG block, all long messages that are destined for a CRT terminal. It also uses this map to build scroll segments of either full screen or screen minus two lines for transmission to the CRT terminals.

**Hard Copy Driver**

This program adjusts output messages (AM0SG or UI0OM) supplied by applications programs for transmission to hardcopy terminals.

**Modify Scroll Display Program**

This program edits move scroll messages supplied by the terminal operator and sets the parameters needed by UIS to build the requested scroll segment for transmission.

**Repeat Last Message Program**

This program processes the terminal operator request for a repeat of the last message sent.

**Canned Message Driver**

This program will build output messages as per the request of applications programs.

## Associated Programs

- 3270 Simulation
- Format Output Message
- Message Switching Input Message Assembly
- Long Message Transmitter.

## Relationships to Other Programs

## Packages Used by This Package

**Basic Message Switching Package**

The XBPS package is used by the UI package to send long messages (more than 98 characters) to the 1977 terminals.

**Format Output Message**

FMSG is used by the UI package to build an output message consisting of a user-supplied output and the associated input message. This is done only for certain types of input messages.

**3270 Input Simulator**

If the input message originated from a 3270 type terminal, the 3270 Input Simulator (CSI) is used to put message in MI0MI format before passing control to the application.

**1977/1980 Output Simulator**

**3270/4505 Output Message Simulator**

If the response is to be sent to a 3270 type terminal the appropriate simulation program must be entered to format the message for 3270 (CSH for hardcopy or CSO for CRT output).

## Packages Using This Package

All programs sending applications output messages may use this package. All programs receiving 1977, 2915, 4505, 3270 (Reservations) inputs receive them from this package.

# Basic Message Switching Package

The message switching function was originally developed as an integral part of the airline reservations application. This chapter describes the main line message switching programs that are maintained in the system. For more information about the message switching application, see “Message Switching” on page 87.

## Description

Certain hardcopy terminals in the system are designated as message switching terminals. This means that they may communicate with each other using message switching as a routing vehicle. The format of input and output messages is described in “External Input and Output Related to This Package” on page 157.

## Purpose

The purpose of the XBPS package is to provide programs which assemble, queue, transmit and log Message Switching messages (XM0RL). It sends unsolicited/overlength reservation messages to hard copy terminals.

## Functions

1. To keypoint core tables in order to provide the system with the latest status at Restart time.
2. To assemble low-speed and high-speed message segments into complete messages.
3. Perform certain editing and validating functions:
   a. Check for excessing repetitive character sequence.
   b. Check for Start of Address (SOA) in first message segment.
   c. Deletion of heading and ending sections on input messages according to ATA/IATA interline message format.
   d. Creation of answerbacks (Date-time and sequence number message indicating input validly received) to message switching terminals.
   e. Validating of addresses in the address section of the message.
   f. Input logging of messages (Customer’s option).
4. Routing of messages to message switching terminal(s) and to the reservations programs as specified in the address section of the message.
5. Interception (temporary withholding) of traffic to a terminal or line.
6. Transmission of message switching messages to low-speed and high-speed terminals.
7. Logging of messages sent to message switching terminals.
8. Transmission of overlength/unsolicited reservation traffic to hard copy terminals.
9. Perform clean-up functions necessary in Restart in order for the system to continue transmitting traffic with a minimum of lost messages after system outage.

## Component Programs

### Programs In This Package

***Restart Program:*** This program is activated after a system RESTART has taken place and the system is being cycled up from 1052 state (see *TPF Main Supervisor*

*Reference*). Its functions can be divided into two categories: those that affect the main-line programs and those that affect the service programs.

The functions performed for the main-line programs are:

1. Update output queue (XU2TQ) by scanning thru the (XT0CB) transmission control blocks.
2. Clear and reset tables in order to be able to start traffic. Tables cleared are XOCT, XTRT and XLAD.
3. Release file addresses and clear the XN1XN entries which were used for assembly of messages for high speed terminals.
4. Release file addresses and clear the XV1XV table to allow input to start properly.
5. Issue CRETC to XLMT in order to provide message assurance of high-speed output.

The functions performed for the service routines are:

1. Complete any retrieval functions which were interrupted previous to RESTART. This means moving retrieval entries from XH0XH, using the XQ1XQ table, onto the transmission control blocks (XT0CB).
2. The Purge Table (UV3RP) is checked to see if the Purge Program (XPPP) was in progress when the system went down. If so, the XRPU will be updated to indicate that a hardware error occurred and another tape must be re-created. The tape is then closed, so that the operator can reinitiate the purge process.
3. The Retrieval Table (UV1RP) is checked to see if the Tape Retrieval Program (XRPP) was in process when system went down. If so, the XRPT will be updated by adding the Process Queue Table (UZ1PQ) onto the front of the Batch Queue Table. The tape will be closed, leaving it up to the operator to restart tape retrieval.

***Input Message Assembly Program:*** This program assembles and checks the format of all incoming communication messages to the Message Switching System. Incoming messages are checked against the format requirements specified in AM0SG, XM0RL and the ATA-IATA Standard Format.

The Physical Line Conversion Table (XP1XP) and the Output Control Table (XW1OC) are used to determine the type of terminal and accumulate statistics on data received by line.

High-speed messages are assembled into 381 bytes blocks using the High Speed Input Status Table (XN1XN). All messages requiring more than one 381 byte block are assembled using the Message Segment Count Table (XE1SC) and the Input Control Table (XV1XV) in order to chain the segments into one complete message. During assembly of the message, XIMA checks for overlength messages, excessive repetition of a character and missing carriage return-line feeds which would cause overprinting at destination terminals.

Heading and/or Communications Control Information (CCI) Sections are deleted from all messages. Messages lacking the required delimiters are sent to the Message Reject Program for transmission to the originator for possible correction.

Completely assembled messages which have been checked are transferred to the Header Analysis Program.

***Header Analysis Program (XHAP):*** XHAP is entered from:

- XIMA with assembled message switching messages.
- XMRJ with corrected reject messages.
- Reservations Programs through the use of the SENDT macro with messages destined for Host Airline Message Switching terminals.

This program checks the address section of each message by verifying each destination mnemonic up to the end-of-address. Each HA destination in the address section is checked using the mnemonic City Code Index Table (XC1CC) and the Function Code Index Table (XF1FF). Special distribution codes (otherwise known as group codes) are also used as a shorthand method of addressing a particular set of terminals. These are found using the Special Distribution List (XD0LS). Messages with errors in the address section are sent to the Message Reject Program (03-XMRJ) which in turn either transmits them back to the originator for correction or sends them to the Construct Reject Program for transmission to a reject operator.

When all addressees have been found, the message is placed in the appropriate output queues (using the XU2TQ which points to the XT0CB). The Terminal Routing Table (XR1TR) indicates which line or terminal queue should be used while the priority code determines into which of the three priorities the message will be placed.

Just prior to placing the message on the output queue, the receiving terminal is checked to determine if alternate routine is required. This can cause the message to be placed on the alternate terminal's output queue. Messages for a high-speed terminal are intercepted in its output queue.

Answerback messages are sent to half-duplex terminals. This requires the use of the Line Index (XLCI) and Terminal Index (XK1CT) in order to find the originator address. The Input Log Directory (XI0DS) is used to maintain the input message sequence number, data-time received and message file address (for input retrieval purposes) when the input logging option is used in the system. Messages from the reservation programs are not logged on the XILD.

When the first HA destination contains an "XC" function, the message is treated as a service message (e.g. Retrieval Request) and control is transferred to the Service Message Dispatcher Program (XSMD).

When it finds the output queues empty, XHAP enters either the Output Message Assembly and Transmission Program for low-speed message transmission or the Long Message Transmitter Program for high speed message transmission upon adding a message entry into the output queue.

***Output Message Assembly and Transmission Program:*** This (XOMA) program controls the removal of messages from output queues (using the XU2TQ and XT0CB).

XOMA is first initiated when a message is added to an empty queue. The queue number used to access the XTQC table is used to access the Output Control Table (XW1OC). The XOCT Table maintains control of the output transmission of messages.

Once transmission of traffic is started, the Communication Control Programs (CCP) return to XOMA after sending output. This continues until XOMA sends all messages in the output queue for that line.

An output message is logged after complete transmission and time stamped with the output terminal sequence number and data/time of transmission. This program enters Log Output Message Program to perform this function when an end of message is reached.

The output queue is updated after a message has been completely transmitted. Chained Transmission Control Blocks (XTCB) are advanced to the primary XTCB record (fixed file address) whenever the primary XTCB record is exhausted.

***Output Log Program:*** This program is divided into two sections: one which creates a CCI section for Message Switching messages and the second which adds messages onto the output queues.

1. Create CCI section for messages (segment XOAA)

   This program segment is entered by XOMA, and XLMT to create CCI sections for messages.

   The program creates the Communications Control Information Section (CCI) for each message. It will create a special CCI for retrieval, monitor and possible duplicate messages (PDM) using the second XTCB entry of a message. Normal messages will have their CCI created from the Output Log Directory (XL0DS). The XOLD is updated and stamped with the date/time of transmission. The XOLD entry number (for message sequence number) and the date/time stamp are used to create the CCI.

   The XTCB being used will be rewritten with a bit set equal to one (restart bit XT0IN2-bit 0) to indicate that the message had been sent. This is used by the Restart Program (03-XRST) to reset the output queues to their latest values to avoid sending a message twice but assure that it will be sent.

   Messages destined for terminals on intercept are removed from the queue and transferred to the Intercept Program (XINT).

2. Add messages to output queues (segment XOBB).

   This segment is used by the following programs:

   **Restart Program**
   To add messages to output queues when correcting unfinished retrieval queues.

   **Output Log Program**
   To add messages onto the output monitoring queues.

   **Message Reject**
   To add messages onto the output queues to be sent back to the originator.

   It is also used by application programs XRET, XRPD, and XSRP to add other message types to queues.

## Relationship To The Long Message Transmission Program

When originally developed XLMT was conceived as part of the message switching function. However, it is now an essential component of the Communication Control Program's management of hard copy terminals. The XLMT program queues output messages on file and segments them for transmission to hard copy terminals whose addresses are referenced by LN/IA/TA - High Speed Line Number, Interchange Address, Terminal Set Address. The following types of messages are transmitted by XLMT via the SLMTC macro:

1. Unsolicited output messages.

2. Overlength solicited output messages, i.e. those requiring segmentation prior to transmission.
3. Output message switching traffic to 1977 terminals.
4. Output traffic to the 1052 or 3215 Operator's Console, the 1980- 21/24 and 3284/3286 Printer Terminals, and the 2980-9 Printer/Keyboard Terminals.

The necessity for XLMT program stems from the diversity and large number of Operational Programs and the impracticality of having each control its own output since transmission to 1980, 3284/3286 1977 terminals initiates Answerback Messages to the processor verifying successful transmission. Messages to 1977 terminals are transmitted in 100- character segments. Messages to 1980, 3284/3286 remote terminals are transmitted in up to 340-character segments (maximum transmission length is dependent upon terminal buffer capacity). Messages to 3284/3286 local terminals are transmitted in one long send.

There are extensive interfaces between the message switching function and XLMT.

## External Input and Output Related to This Package

### Input

Action code Y, Standard message switching message entries e.g.:

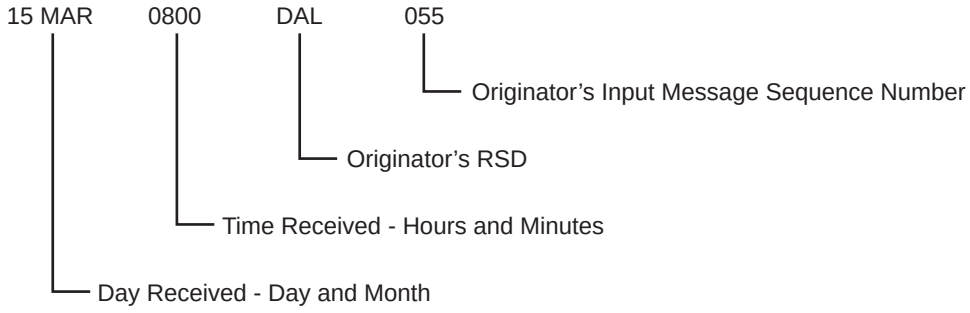
```
Y

QU NYCMMQT. BOSMMQT 091230

THIS IS A SINGLE DESTINATION MESSAGE TO CITY - NYC, FUNCTION-MM,
AIRLINES QT.
```

The entry is sent to 03-UIL, the Long Message Assembly Program where it is passed on to this package. Other messages of this type with the same action code will be accepted. They can be sent to other terminals providing the city and/or function codes are correct.

### Output

Response for a valid message is called an answerback.

For Example:




The input message will be forwarded to the terminal designated as NYCMM (city-function code).

Response for an invalid message is called reject. In the example, this occurs in QT airlines if NYC was not a valid city code:

## Internally Activated Processes

1. A CRET is issued by the XRST program in this package to start high-speed message switching traffic to those line(s) and terminal(s) which have been removed from intercept.
2. Both XOMA (low-speed output) and XLMT (high-speed) can be initiated by several programs. See XOLG since it does the queueing of output for most of these programs.
3. The Relieval Program (XREL) can initiate either XOMA or XLMT when message queues are relieved from intercept.

# Index

File Number: S370/30XX-30
Program Number: 5748-T14

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SH31-0145-03