IBM

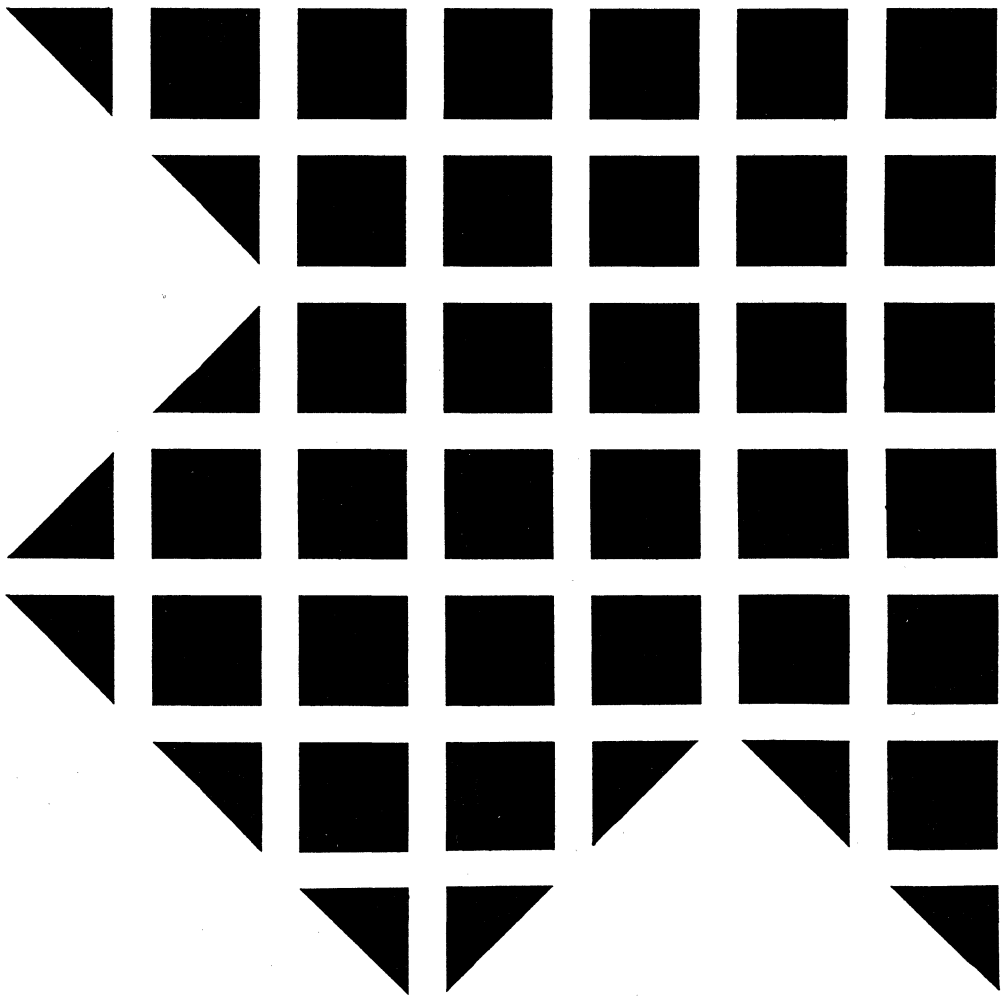
Virtual Machine/Extended Architecture™
System Product

GC23-0503-0

VM/XA™ SP Release 2

Serviceability Enhancements Program Update Information

APAR VM37518



Virtual Machine/Extended Architecture™ System Product

GC23-0503-0

VM/XA™ SP Release 2

Serviceability Enhancements Program Update Information

APAR VM37518

First Edition (September 1989)

This edition applies to Release 2 of the Virtual Machine/Extended Architecture System Product (VM/XA SP) Licensed Program 5664-308. This edition, documenting Serviceability Enhancements between-release support information, is a supplement to *VM/XA SP Installation and Service*, SC23-0364, and to the *VM/XA SP System Messages and Codes Reference*, SC23-0376. Technical changes and additions to the text and illustrations of these books are indicated by a vertical bar to the left of each change. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to International Business Machines, Department 52Q/MS 458, Neighborhood Road, Kingston, N.Y. 12401. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

Purpose

This publication applies to the Serviceability Enhancements Support Supplement to VM/XA System Product Release 2 (APAR VM37518). It explains how to service your VM/XA System Product system. Use it to supplement *VM/XA SP Installation and Service*, SC23-0364, and the *VM/XA SP System Messages and Codes Reference*, SC23-0376.

Audience

This publication is intended for those people responsible for installing and applying service to the VM/XA System Product Release 2.

How to Use This Publication

Use this book in conjunction with the *VM/XA SP Installation and Service*, SC23-0364. The information in this book **replaces** the Table of Contents, Chapters 6 through 20, and Appendixes A through I, and adds a new appendix, Appendix J.

Technical changes and additions to the text and illustrations are indicated by a vertical bar (|) to the left of the change.

There are no programming interfaces defined in this publication.

Contents

Part 2. Servicing the System

Part 2 of this book contains the following chapters:

- Chapter 6, "VM/XA™ Sytem Product Service—An Overview" on page 355, which provides an overview of the service tasks that you may need to perform. It also contains a description of the MAINT virtual machine (the user ID that you use to perform the service), and a list of the tools and EXECs you use to apply service.
- Chapter 7, "How VM/XA System Product Uses Control Files and Update Files" on page 377, which describes what control files and update files are, and how VM/XA SP uses them
- Chapter 8, "Files Used in Program Update Service and Corrective Service" on page 385, which describes the program update tape, the corrective service tape, and the files used in applying program update service or corrective service to VM/XA SP
- Chapter 9, "Receiving Program Update Service or Corrective Service for CMS" on page 435, which provides a procedure for receiving program update service or corrective service for CMS
- Chapter 10, "Receiving Program Update Service or Corrective Service for CP" on page 443, which provides a procedure for receiving program update service or corrective service for CP
- Chapter 11, "Applying Program Update Service or Corrective Service to CMS" on page 449, which provides a procedure for applying program update service or corrective service to CMS
- Chapter 12, "Applying Program Update Service or Corrective Service to CP" on page 453, which provides a procedure for applying program update service or corrective service to CP
- Chapter 13, "Rebuilding CMS after Applying Service" on page 457, which provides a step-by-step procedure for rebuilding CMS after applying program update service, corrective service, or local service
- Chapter 14, "Rebuilding CP after Applying Service" on page 515, which provides a step-by-step procedure for rebuilding CP after applying program update service, corrective service, or local service
- Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility" on page 537, which outlines a procedure for receiving and applying program update service or corrective service to the dump viewing facility and for rebuilding the dump viewing facility
- Chapter 16, "Program Update Service or Corrective Service to GCS" on page 547, which outlines a procedure for receiving and applying program update service or corrective service to the group control system and for rebuilding the group control system
- Chapter 17, "Program Update Service to Licensed Programs" on page 563, which outlines a procedure for receiving and applying program update service to products that have a product service EXEC rather than a product parameter file
- Chapter 18, "Receiving and Applying Local Service" on page 565, which provides a step-by-step procedure for receiving and applying local service to VM/XA System Product
- Chapter 19, "Emergency Local Service Using the Patch Facility" on page 577, which provides a step-by-step procedure for applying patches to the CP or CMS nucleus
- Chapter 20, "Removing Service from VM/XA SP" on page 581, which describes how to remove service from VM/XA System Product.

VM/XA SP is a trademark of the International Business Machines Corporation

Chapter 6. VM/XA System Product Service—An Overview

This chapter contains:

- An introduction to service
- An overview of program update service
- An overview of local service
- A description of using the MAINT virtual machine for service
- A description of the tools and EXECs used in applying local service.

Introduction

The VM/XA system as installed by the customer consists of several products, such as VM/XA System Product, PROFS, and FORTRAN. VM/XA System Product is a product which consists of the components CP, CMS, GCS, and dump viewing facility. Each component consists of numerous parts, such as macros, copy files, assembler modules, text decks, executable modules, and EXECs. Some parts are serviced by complete part replacement only. These parts are sometimes called **object-maintained** code. Other parts are serviced by combining an update with the base part. These parts are sometimes called **source-maintained** code.

Parts serviced by replacement require the addition of a replacement part to the system. Parts serviced by update require the careful application of an update file to the existing control structure.

Each problem that requires a fix from IBM is assigned an authorized problem analysis report (APAR) number. (The term "APAR" is ordinarily used to refer to both the fix and the report. In this book, "APAR" means the fix.) An APAR may contain service to more than one part. It may contain update service, replacement service, or both.

A program temporary fix (PTF) is a package of APARs. For update service, the PTF contains only one APAR, because each update file is a change, or "delta", to the base part. Each problem is fixed by the application of a separate delta. For replacement service, the PTF may contain more than one APAR, because replacement parts must include all previous APAR fixes in addition to the APAR fix for the problem being solved by the replacement. Figure 5 on page 356 shows three PTFs being applied to the system. PTF 1 and PTF 3 each contain a single APAR consisting of an update to Part A. PTF 2 is more complex. It contains three APARS: APAR V, APAR W, and APAR X. All three APARs require adding a replacement part to Part C. APAR X also includes updates to Part A and Part B. (PTF 2 is cumulative. Two earlier PTFs are also available, one containing only APAR V and one containing APAR V and APAR W.)

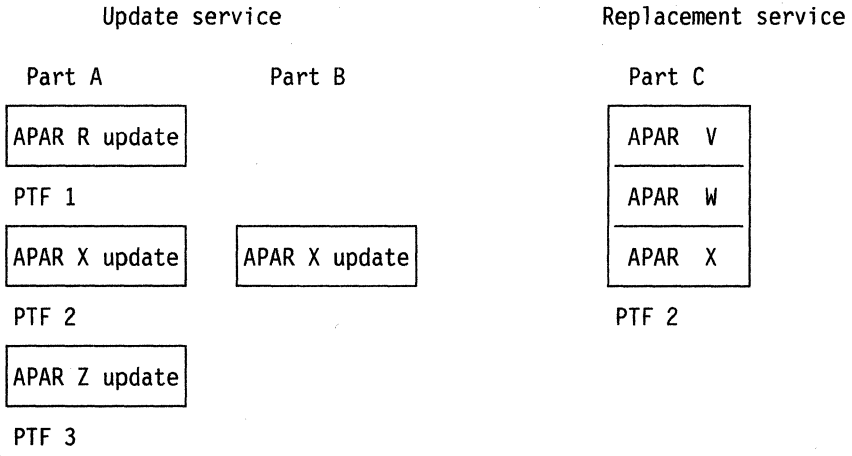


Figure 5. Update Service and Replacement Service

Service is the process of applying fixes—PTFs to whenever they are available to the system after installation. Problems fall into these categories:

- A PTF is available on the latest program update tape (PUT).
- A PTF is available, but no PUT is available.
- No PTF is available.

These circumstances determine the format in which the service is supplied. This, in turn, decides the procedure for applying it. These procedures are:

- Program update service:
  - A PTF is available.
  - The PTF is supplied on a PUT tape.
  - The PTF is automatically applied.
  - Application may be preventive (massive application) or selective.
- Remedial service, to be replaced by program update service when a PUT becomes available.
  - Corrective service:
    - A PTF is available, but no PUT is available.
    - The PTF is supplied on a corrective service tape.
    - The PTF is automatically applied.
  - Local service:
    - No PTF is available.
    - The service is originated by the customer, or is supplied by IBM on a tape other than a PUT tape or corrective service tape, in hard copy, or over the phone.
    - The service is manually applied.
    - The service may correct the problem or circumvent it (branch around or otherwise avoid the code in which the problem occurs).

Note: These classifications have nothing to do with the classification of service as update service or replacement service. Program update service, corrective service, and local service can all require either updates or replacement.

Both massive and selective application of PTFs on the PUT use the same procedure, called program update service because its input comes from the program update tape. Massive application of PTFs on the PUT is intended to avoid problems. For this reason, program update service is sometimes called preventive service. Selective application of PTFs on the PUT is a response to a problem after it occurs.

When a problem occurs for which a PTF does not exist on the latest PUT, IBM will provide a corrective fix if the problem is known and the PTF is ready for the next PUT. Corrective service comes on a corrective service tape with the same format as the program update tape. It is applied in essentially the same way as program update service.

If the problem is not known, IBM may be able to provide a temporary circumvention over the phone for a severe problem. (Circumventive service may also be provided on tape.) The circumvention usually does not fix the problem, but prevents failure by disabling the failing function. In this book, service that does not come from the PUT tape or a corrective service tape is called local service, even when you get it from IBM.

Local service for replacement parts is like the procedure used for update parts, except that you do not need to assemble replacements for object code. Once updates are applied and assemblies are completed, update parts and replacement parts are the same.

For parts on replacement service there are two circumventive procedures. One procedure, the patch facility, is for the CP nucleus and CMS nucleus. The other procedure uses ZAP for CP utilities and command modules not in the nucleus.

Program Update Service Overview

All IBM customers have a Customer Profile at IBM Software Distribution, or ISD (formerly known as PID). This profile lists all program products for which a customer has licenses. ISD sends out program update service tapes which are tailored to the Customer Profile—this is how you receive program update service for those program products for which you have licenses.

Program update service is sent out to VM/XA SP customers either on a regular basis or when you request it. It arrives in the form of a Program Update Tape (or PUT), which you apply automatically using a set of EXECs. The PUT, which may be one or more tape volumes, contains program update service files for one or more program products.

The PUT is cumulative. All update files up to the current level are included. For parts serviced by updates (source-maintained code), the latest text deck, which includes all APARs up to the current level, is also shipped. For parts serviced by replacement (object-maintained code), a deck is shipped for each PTF and will be reshipped for several PUT cycles.

The VMFREC EXEC is used to load program update service for each VM/XA System Product component to the appropriate DELTA disk as specified in the product parameter file (PPF). The VMFAPPLY EXEC is used to apply it and to build the AUX control structure on the specified APPLY disk. The VMFBLD EXEC is used to build your nucleus after you receive and apply service. These EXECs are supplied on the VM/XA System Product product tape.

For products other than VM/XA System Product, the service files include a CMS service EXEC for each program product on the PUT. The service EXEC is especially designed for that program product. VMFREC maps the PUT and calls the appropriate service EXEC for each product, other than VM/XA System Product, being serviced.

The PUT also contains an EXEC called VMSERV, for compatibility. For products other than VM/XA System Product, you can use VMSERV as an alternative to VMFREC. Mount the tape on a tape drive, load the first tape file under CMS, and invoke the VMSERV EXEC. VMSERV supervises the installation of program update service for the program products you have.

The format of each volume of the PUT is as follows:

- **The first tape file** contains the PUT DOCUMENT, the VMSERV EXEC, and the latest tested level of the service EXECs (with PTF-numbered file names).
- **The second tape file** contains a Memo to Users for each program product service on the volume, and the latest tested level of the product parameter file for each product on the tape.
- **The remaining tape files** contain service files for the program products on that volume.

When you invoke the VMFREC EXEC with the INFO (MEMOS option, you can load the PUT document and all memos to users to disk and print them. You must read the instructions on the PUT document and the Memo to Users before you continue program update service installation.

Specific instructions for receiving and applying program update service and for rebuilding components afterwards are found in Chapter 9 through Chapter 17.

Corrective Service Overview

Corrective service is service IBM sends you to correct a specific problem that you or another customer has encountered and reported. It is sent on a corrective service tape, which has the same format as the program update tape. It is applied the same way as program update service (see "Program Update Service Overview" on page 357), except that service is loaded and applied to the LOCAL1 disks instead of the DELTA1 and APPLY disks. This is accomplished by redefining the DELTA1 and APPLY disk strings to be the same as the LOCAL1 disk string in the override section of the product parameter file. (See "The Product Parameter File" on page 401.)

Specific instructions for receiving and applying corrective service and for rebuilding components afterwards are found in Chapter 9 through Chapter 16.

Local Service Overview

Local service is any service that is not supplied on a program update tape or corrective service tape. It can be service that you originate, or service that is sent to you by IBM to correct or circumvent a specific problem that you have encountered and reported. Circumventive service is sometimes read to you by an IBM service representative over the telephone.

Changes to the system are provided as new parts, replacement parts, or changes (updates) to existing parts initially shipped on the product tape. They include:

- Updates to an assembler source file (filetype is ASSEMBLE), macro (filetype is MACRO), control block (filetype is COPY), EXEC (filetype is $EXEC), or XEDIT macro (filetype is $XEDIT)
- Replacement of existing object code (filetype is usually TXT*nnnnn*), EXECs (filetype is EXEC), XEDIT macros (filetype is XEDIT) HELP files, TXTLIB files, or MODULE files
- Patches to a text file (filetype is usually TXT*nnnnn*) for which there is no assembler source file.

You will perform the following general steps manually for local maintenance:

1. Receive the following files to the alternate LOCAL1 disk:
   - Source update files
   - Replacement files.

2. Apply the fixes by creating auxiliary control files on the alternate LOCAL1 disk.
3. Build these changes into a alternate running system on the new BUILD1 disk.

Go to Chapter 18, "Receiving and Applying Local Service" on page 565 and follow the instructions for applying local service. Each chapter also gives instructions on how to build, test, and finally save your new system.

The MAINT Virtual Machine

By convention, the MAINT virtual machine is used to perform installation and service tasks. (Your installation may use a different virtual machine.) MAINT's directory entry is part of the sample file, USER DIRECT, that is supplied with the starter system. The sample directory is printed in Appendix C, "VM/XA System Product Starter System Information" on page 747. You may wish to alter MAINT's virtual machine configuration, but remember that IBM created the configuration with the service process in mind.

Minidisks form an important part of MAINT's configuration. The strategy IBM follows is to separate key files onto different minidisk strings. A **string** is a set of minidisks defined in the product parameter file (see "The Product Parameter File" on page 401) for a particular use. The different strings defined in the product parameter file are:

String	Use
TASK	Files to be accessed after the A-disk and before the component data base (for example, a service execs disk)
BASE1	Object code
BASE2	Source code
DELTA1	Updates and replacement parts from the program update tape
APPLY	Auxiliary control files generated by the VMFAPPLY EXEC
LOCAL1	Updates, replacement parts, and auxiliary control files for corrective service and local service
BUILD1	The system built from the base code on the BASE disks plus the updates on the DELTA1 and LOCAL1 disks
BUILD2-*n*	System disk extensions—exact use varies by component. (For CMS, the HELP files are here.)
SYSTEM	CMS-related files

When there is more than one minidisk in a string, you can use one for your current system and one, holding files from previous service levels, for an alternate system.

MAINT's service minidisks are described in Table 10 on page 361.

Table 10 (Page 1 of 3). MAINT's Minidisks for System Service

Comp.	String	Disk	Contents
CMS, DV	LOCAL1	0395	CMS corrective service DELTA disk. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
		0691	Minidisk with customized files for CMS and samples for DMSNGP and local service. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
		0391	Cumulative merge of previous levels of CMS LOCAL1 disks. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
	APPLY	0692	New CMS control disk containing CMS AUX files which have been generated from text files by VMFAPPLY
		0392	Cumulative merge of previous levels of CMS APPLY disks
	DELTA1	0593	CMS text files or update files containing program temporary fixes (PTFs—also known as CMS service files) and auxiliary control files (AUXXA files) from the PUT
		0293	Cumulative merge of previous levels of CMS DELTA1 disks
	BASE1	0193	CMS text file retention, CMS generation and utility tools, IOCP, and the dump viewing facility
	BASE2	0393	CMS source file retention, macro definitions, and control blocks
	BUILD1	0490	New CMS system disk containing CMS modules, control file, and CMS MACLIBs
	BUILD2	049D	New mixed-case American English HELP files for CP, CMS, and GCS
	BUILD3	0501	New CMS system disk for EREP files
	BUILD4	049C	New uppercase American English HELP files for CP, CMS, and GCS
	SYSTEM	0191	MAINT's work minidisk
		0295	CP alternate LOCAL1 minidisk
		0591	CP intermediate alternate LOCAL1 minidisk
		0291	CP current LOCAL1 minidisk
		0594	CP alternate DELTA1 minidisk
		0294	CP current DELTA1 minidisk
		0194	CP current BASE1 minidisk

Note: For the CMS and DV components, the minidisks on the SYSTEM string are required for access to CP's MACLIBs.

Table 10 (Page 2 of 3). MAINT's Minidisks for System Service			
Comp.	**String**	**Disk**	**Contents**
CP	LOCAL1	0295	CP corrective service DELTA disk. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
		0591	Customized files for CP, HCPBOX, HCPRIO, HCPSYS, and USER DIRECT files, and local service. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
		0291	Cumulative merge of previous levels of CP LOCAL1 disks. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
	APPLY	0492	New CP control disk containing CP AUX files which have been generated from text files by VMFAPPLY
		0192	Cumulative merge of previous levels of CP APPLY disks
	DELTA1	0594	CP text files or update files containing program temporary fixes (PTFs—also known as CP service files) and auxiliary control files (AUXXA files) from the PUT
		0294	Cumulative merge of previous levels of CP DELTA1 disks
	BASE1	0194	CP text file retention, CP load list and control file, and CP MACLIBs
	BASE2	0394	CP source file retention, macro definitions, and copy files
	BUILD2	049D	New mixed-case American English HELP files for CP, CMS, and GCS
	BUILD3	049C	New uppercase American English HELP files for CP, CMS, and GCS
	BUILD4	0490	New CMS system disk containing CMS modules, control file, and CMS MACLIBs
	SYSTEM	0191	MAINT's work minidisk
		0395	CMS alternate LOCAL1 minidisk
		0691	CMS intermediate alternate LOCAL1 minidisk
		0391	CMS current LOCAL1 minidisk
		0593	CMS alternate DELTA1 minidisk
		0293	CMS current DELTA1 minidisk
		0193	CMS current BASE1 minidisk
Note: For the CP component, the minidisks on the SYSTEM string are required for access to the service EXECs. If you set up a service EXECs build disk on the TASK disk string, you can remove all but the 191 disk from the SYSTEM disk string.			

Table 10 (Page 3 of 3). MAINT's Minidisks for System Service

Comp.	String	Disk	Contents
GCS	LOCAL1	0495	GCS corrective service DELTA disk. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
		0491	Current GCS local service disk. This disk contains the cumulative merge of both the 495 and 791 from previous levels. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
		0791	New GCS local service disk. This disk is treated as part of the LOCAL disk string during PUT service and as part of both the DELTA and APPLY strings during corrective service.
	APPLY	0592	GCS current cumulative merge of AUX files which have been generated by VMFAPPLY and UPDATE shells created by VMFREC
		0892	New GCS AUX files which have been generated by VMFAPPLY and UPDATE shells created by VMFREC
	DELTA1	0596	GCS current cumulative merge of text files, update files and AUX files from previous levels of PUT
		0896	New GCS text files, update files and AUX files
	BASE1	0595	GCS text file retention, GCS load list, control file, and GCS MACLIBs
	BUILD1	0895	New GCS system
	BUILD2	089E	New GCS system extensions
	BUILD3	049D	New mixed-case American English HELP files for CP, CMS, and GCS
	BUILD4	049C	New uppercase American English HELP files for CP, CMS, and GCS
	BUILD5	0490	New CMS system disk containing CMS modules, control file, and CMS MACLIBs
	SYSTEM	0191	MAINT's work minidisk
		0395	CMS alternate LOCAL1 minidisk
		0691	CMS intermediate alternate LOCAL1 minidisk
		0391	CMS current LOCAL1 minidisk
		0593	CMS alternate DELTA1 minidisk
		0293	CMS current DELTA1 minidisk
		0193	CMS current BASE1 minidisk

Note: For the GCS component, the minidisks on the SYSTEM string are required for access to the service EXECs. If you set up a service EXECs build disk on the TASK disk string, you can remove all but the 191 disk from the SYSTEM disk string.

MAINT's minidisk configuration allows a multilevel updating scheme that preserves the data integrity of the source code and allows you to build a system with the latest updates. The organization of MAINT's minidisks is shown in Figure 6 on page 364. This layout allows you to build a new system on the alternate minidisks without altering your running system and lets you back out to the old system if necessary. When you are satisfied with your new system, the alternate disks are copied to the current system's disks so that the alternate disks can be used to build your next system.

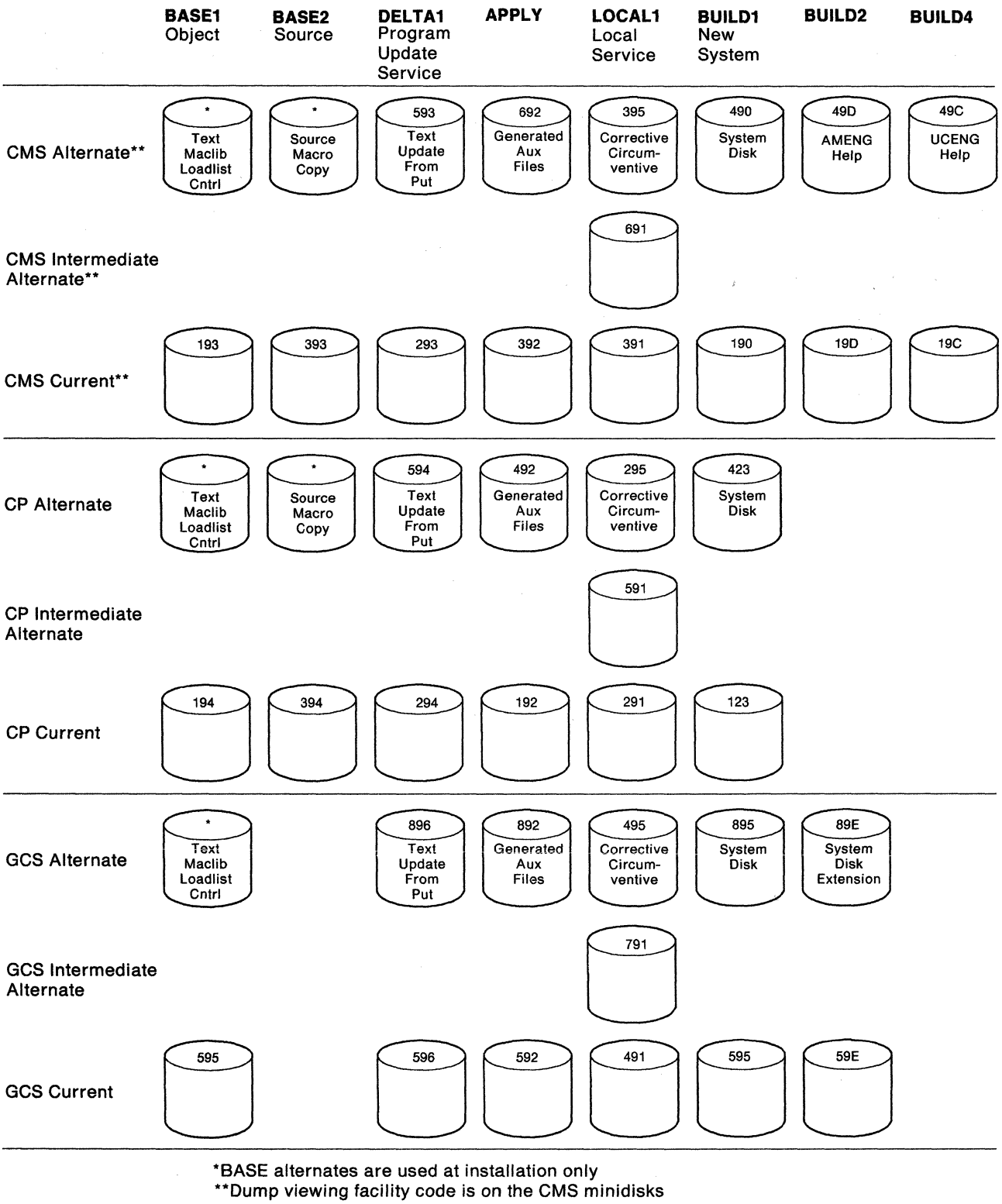


Figure 6. Alternate, Intermediate Alternate, and Current Minidisks for System Service

The disk layout in Figure 6 on page 364 is created on the following principles:

1. Never change anything IBM sends you. To be sure that you don't, never write-link to a disk you don't want to write on. The BASE disks are read-only during service. The DELTA disks are read-only except when VMFREC and VMFAPPLY are running.
2. Keep components separate from each other, so that (for example) changes to CP will not affect CMS.
3. Keep locally applied service and corrective service separate from PUT service, so that changes to the LOCAL disk will not inadvertently change IBM service on the DELTA disk.
4. Keep base release installation separate from service, so that changes to the LOCAL disk or VMFREC will not inadvertently change the original product as installed on the BASE disks.
5. Keep alternates for LOCAL and DELTA disks so that you can back out.

   Plan for backout in case something goes wrong when you apply service. Alternates make reconstruction of the previous level easier.
6. Keep an alternate BUILD disk.

   Don't change the system while it is running. An alternate BUILD disk minimizes the amount of reconstruction necessary to back out.

Figure 7 shows the system disks accessed during the process of applying service to CP and CMS. This is done while the system is running in a CMS environment. The current system disks are not changed by service and remain accessed at all times. The service process builds the new system on the alternate system disks, which will be copied to the current system disks to make room for the next service cycle when the service process is complete and stabilized.

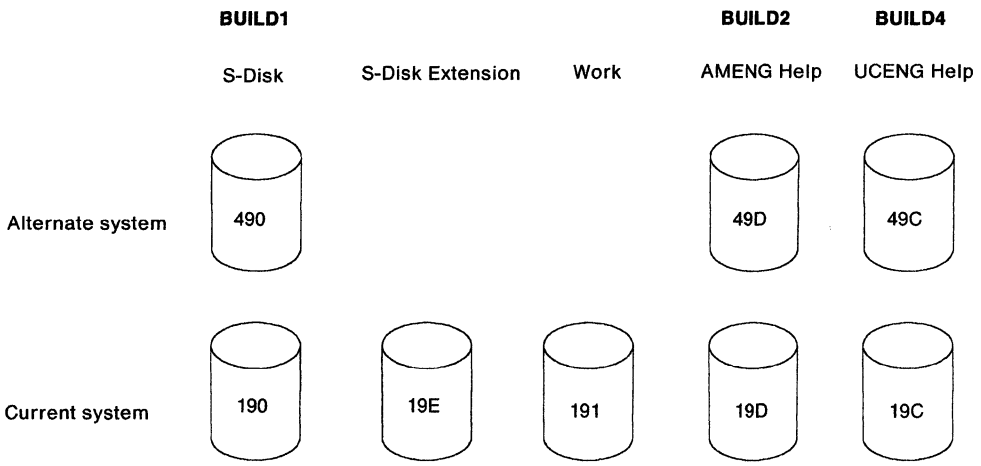


Figure 7. Minidisks for Running CMS

The system alternate disks are the BUILD disks in Figure 6 on page 364. They are shown here to emphasize the fact that the service process builds on a set of disks separate from the running disks.

Tools and EXECs Used During the Service Process

You need to use many different programs and EXECs during the service procedure. Below is a list of the programs and EXECs you will use:

VMFREC Use VMFREC to load product service from a tape to your system. This EXEC is required for VM/XA System Product service. It is compatible with VMSERV and existing product service EXECs. You can use it to load the tape document and all Memos to Users for that tape as well as to map the tape. The map is compatible with VMSERV.

For more information on VMFREC, see "VMFREC EXEC" on page 709.

VMFAPPLY Use VMFAPPLY with the APPLY list to create AUX files for those PTFs being added to your system.

For more information on VMFAPPLY, see "VMFAPPLY EXEC" on page 672.

VMFMAC The VMFMAC EXEC updates and rebuilds macro libraries.

For more information on VMFMAC, see "VMFMAC EXEC" on page 688.

VMFNLS The VMFNLS EXEC updates national language-related files, for example, message repositories.

For more information on VMFNLS, see "VMFNLS EXEC" on page 698.

VMFHASM VMFHASM invokes the UPDATE command to automatically incorporate update files for a particular ASSEMBLE file into a temporary copy of the ASSEMBLE file, then invokes the H Assembler to assemble the copy. The assembled files (called TEXT files) are used when the system is regenerated by the VMFBLD EXEC.

For more information on VMFHASM, see "VMFHASM EXEC" on page 684.

VMFBLD Use VMFBLD to build the nucleus parts of VM/XA System Product.

For more information on VMFBLD, see "VMFBLD EXEC" on page 679.

VMFOVER The VMFOVER EXEC makes a temporary copy of one component section from the product parameter file with changes made in accordance with the override section of the product parameter file or a separate PPF override file. The other service and installation EXECs use this copy instead of the base product parameter file.

For more information on VMFOVER, see "VMFOVER EXEC" on page 703.

VMFPLC2 VMFPLC2 loads all or a specified number of files from tape. Directions for its use as a command appear in the PUT document; it is also automatically invoked by VMFREC, VMSERV, and product service EXECs.

For more information on VMFPLC2, see "VMFPLC2 Command" on page 705.

VMFSETUP The VMFSETUP EXEC is invoked by the other service EXECs to set up the minidisk access order.

For more information on VMFSETUP, see "VMFSETUP EXEC" on page 722.

VMFVIEW The VMFVIEW EXEC invokes XEDIT to allow you to view the exception logs. Using the VMFVIEW EXEC's PF key assignments, you can view all the messages of a specific type, all the messages of a specific number, the HELP screen for a particular message, and move backwards and forwards through the displayed exception log.

For more information on VMFVIEW, see "VMFVIEW EXEC" on page 724.

VMSERV VMFREC has superseded VMSERV, but you may continue to use the VMSERV EXEC to install products other than VM/XA System Product from the PUT. This EXEC is in the first file on a PUT; it can be used to print the PUT document and all memos to users for that PUT.

XEDIT XEDIT is the CMS editor shipped with VM/XA System Product. Use it to create control files in the local service procedure.

If you need information on using XEDIT, please see *VM/XA SP System Product Editor Command and Macro Reference*, or *VM/XA SP System Product Editor User's Guide*.

"MAP 0001: Select a Service Procedure," "MAP 0004: Apply Local Service" on page 370 , "MAP 0003: Apply Program Update Service or Corrective Service" on page 368, and "MAP 0005: Build the System" on page 371 outline the service process, showing when you will need to use the key service EXECs. The service maps also refer you to the exact places in Chapter 9 through Chapter 16 where the major steps are detailed.

Note that the essential differences between applying program update service or corrective service and applying local service are in the manner of *receiving* and *applying* service. For all kinds of service, the system is *built* in the same way.

For program update service of products other than VM/XA System Product Release 2, the product build EXEC is invoked by issuing the VMFBLD EXEC for the product.

MAP 0001: Select a Service Procedure

001

Have you read Chapter 6, "VM/XA System Product Service—An Overview" on page 355, Chapter 7, "How VM/XA System Product Uses Control Files and Update Files" on page 377, and Chapter 8, "Files Used in Program Update Service and Corrective Service" on page 385?

Yes **No**

002

– Read those chapters before you do anything else.

003

Is the service on either a program update tape or corrective service tape?

Yes **No**

004

– Go to "MAP 0004: Apply Local Service" on page 370.

005

– Go to "MAP 0002: Receive Program Update Service or Corrective Service" on page 368.

MAP 0002: Receive Program Update Service or Corrective Service

001

– VMFREC EXEC

CMS Chapter 9, "Receiving Program Update Service or Corrective Service for CMS" on page 435
CP Chapter 10, "Receiving Program Update Service or Corrective Service for CP" on page 443
DV Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility," "Step 2. Receive Service" on page 539
GCS Chapter 16, "Program Update Service or Corrective Service to GCS," "Step 2. Receive Service" on page 549

Are you servicing both CP and CMS?
Yes **No**

002

– Go to "MAP 0003: Apply Program Update Service or Corrective Service."

003

Have you run VMFREC to receive both CP and CMS service?
Yes **No**

004

– Go back to Step 001.

If you are servicing both CP and CMS, you *must* receive service for both components before you apply service to either.

005

– Go to "MAP 0003: Apply Program Update Service or Corrective Service."

MAP 0003: Apply Program Update Service or Corrective Service

001

Are you servicing VM/XA SP?
Yes **No**

002

– PRODUCT SERVICE EXEC

See the documentation for the specific product.

003

(Step **003** continues)

003 (continued)

Are you servicing CP?
Yes **No**

004

– Continue with Step 009.

005

Are you also servicing CMS?
Yes **No**

006

– Continue with Step 009.

007

Have you already applied service to CMS?
Yes **No**

008

– STOP.

If you are servicing both CP and CMS, you *must* apply service to CMS before you apply service to CP.

009

– VMFAPPLY EXEC

CMS Chapter 11, "Applying Program Update Service or Corrective Service to CMS" on page 449
CP Chapter 12, "Applying Program Update Service or Corrective Service to CP" on page 453
DV Chapter 15, "Step 3. Apply the Updates" on page 541
GCS Chapter 16, "Step 3. Apply the Updates" on page 551

Are you servicing both CP and CMS?
Yes **No**

010

– Go to "MAP 0005: Build the System" on page 371.

011

Have you run VMFAPPLY to apply service to both CP and CMS?
Yes **No**

012

– Go back to Step 009.

If you are servicing both CP and CMS, you *must* apply service to both components before you build either.

013

– Go to "MAP 0005: Build the System" on page 371.

MAP 0004: Apply Local Service

001

Is the service in update form?
Yes **No**

002

– Continue with Step 004.

003

– Apply the update manually.

Chapter 18, "Receiving and Applying Local Service," "Step 3. Apply Service" on page 570

004

Is the service either a patch or a ZAP?
Yes **No**

005

– Apply the update manually.

Chapter 18, "Receiving and Applying Local Service," "Step 3. Apply Service" on page 570

– Go to "MAP 0005: Build the System" on page 371.

006

Are you servicing the nucleus?
Yes **No**

007

– VMFZAP EXEC

Appendix B, "EXEC and Command Format Summaries," "VMFZAP EXEC" on page 729

– Go to Step 009 on page 371.

008

– Patch Facility

Chapter 19, "Emergency Local Service Using the Patch Facility" on page 577

009

Are you servicing both CP and CMS?

Yes **No**

010

– Go to "MAP 0005: Build the System."

011

Have you applied local service to both CP and CMS?

Yes **No**

012

– Go back to Step 001 on page 370.

If you are servicing both CP and CMS, you *must* apply service to both components before you build either.

013

– Go to "MAP 0005: Build the System."

MAP 0005: Build the System

001

Are you servicing CP?

Yes **No**

002

– Continue with Step 007 on page 372.

003

Are you also servicing CMS?

Yes **No**

004

– Continue with Step 007 on page 372.

005

(Step **005** continues)

005 (continued)

Have you already built CMS?
Yes **No**

006

– STOP.

If you are servicing both CP and CMS, you *must* build CMS before you build service to CP.

007

Are you servicing macros?
Yes **No**

008

– Continue with Step 010.

009

– VMFMAC EXEC

CMS Chapter 13, "Rebuilding CMS after Applying Service," "Step 1. Build a New CMS Macro Library" on page 458, "Step 2. Build a New CP Macro Library" on page 461
CP Chapter 14, "Rebuilding CP after Applying Service," "Step 2. Build a New CP Macro Library" on page 518
GCS Chapter 16, "Step 4. Build a New GCS Macro Library" on page 552

010

Are you servicing the message repository?
Yes **No**

011

– Continue with Step 013.

012

– VMFNLS EXEC

CMS Chapter 13, "Step 3. Update the CMS Message Repository" on page 464
CP Chapter 14, "Step 3. Update the CP Message Repository" on page 521
GCS Chapter 16, "Step 5. Update the GCS Message Repository"

013

(Step **013** continues)

013 (continued)

Are you servicing an ASSEMBLE file?
Yes **No**

014

– Continue with Step 016.

015

– VMFHASM EXEC *

CMS Chapter 13, "Step 4. Assemble the Changed ASSEMBLE Files" on page 465
CP Chapter 14, "Step 4. Assemble the Changed ASSEMBLE Files" on page 522
GCS Chapter 16, "Step 6. Assemble and Build the GCS Nucleus" (substep 6 on page 556)

016

Are you servicing the nucleus?
Yes **No**

017

– Continue with Step 019.

018

– VMFBLD EXEC

CMS Chapter 13, "Step 5. Build the Nucleus" on page 467
CP Chapter 14, "Step 5. Build the CP Nucleus" on page 523
GCS Chapter 16, "Step 6. Assemble and Build the GCS Nucleus" (substep 11 on page 557)

019

Are you servicing either CP or the dump viewing facility?
Yes **No**

020

– Continue with Step 022.

021

– UTILITY EXEC

CP Chapter 14, "Step 8. Build Utilities" on page 530
DV Chapter 15, "Step 4. Build the Dump Viewing Facility" on page 542

022

(Step **022** continues)

022 (continued)

Are you servicing CMS?
Yes **No**

023

– Continue with Step 025.

024

– CMSGEND EXEC

CMS Chapter 13, "Step 6. Generate Executable Modules" on page 469

025

Are you servicing System Product Interpreter programs (EXECs or XEDIT macros)?
Yes **No**

026

– Continue with Step 028.

027

– EXECUPDT EXEC

CMS Chapter 13, "Step 7. Regenerate System Product Interpreter Programs" on page 474
CP Chapter 14, "Step 9. Regenerate System Product Interpreter Programs" on page 533
DV Chapter 15, "Step 5. Regenerate System Product Interpreter Programs" on page 543

028

– Build test CMS named saved systems.

All Chapter 13, "Step 10. Build Test Named Saved Systems" on page 481

Are you servicing CMS?
Yes **No**

029

– Go to Step 031 on page 375.

030

– Install test saved segments.

CMS Chapter 13, "Step 11. Install Test CMSDOS, CMSBAM, CMSVSAM, and CMSAMS Saved Segments" on page 484, "Step 12. Install Test CMSINST and HELP Saved Segments" on page 491

031

– Test the new CMS system.

All Chapter 13, "Step 13. Test the System" on page 495

Does it work?
Yes **No**

032

– Correct the problem and try again.

033

– Purge CMS test named saved systems and saved segments.

All Chapter 13, "Step 14. Purge the Test Named Saved Systems and Saved Segments" on page 496

– DDR alternate CMS system disks to current CMS system disks.

All Chapter 13, "Step 15. DDR Alternate Disks to System Disks" on page 497

– Rebuild the CMS nucleus (VMFBLD EXEC)

All Chapter 13, "Step 16. Rebuild the Nucleus" on page 499

– Rebuild CMS named saved systems.

All Chapter 13, "Step 17. Rebuild Named Saved Systems" on page 501

Are you servicing CMS?
Yes **No**

034

– Go to Step 036.

035

– Reinstall saved segments.

CMS Chapter 13, "Step 18. Reinstall the CMSDOS, CMSBAM, CMSVSAM, and CMSAMS Saved Segments" on page 504, "Step 19. Reinstall the CMSINST and HELP Saved Segments" on page 510

036

– Back up the new system.

CMS Chapter 13, "Step 20. Back Up the Named Saved Systems" on page 514
CP Chapter 14, "Step 11. Back Up the CP System" on page 536
DV Chapter 15, "Step 7. Back Up the Dump Viewing Facility" on page 546
GCS Chapter 16, "Step 9. Back Up the GCS Named Saved Systems" on page 561

You have finished the service process.

Chapter 7. How VM/XA System Product Uses Control Files and Update Files

This chapter describes:

- What control files, auxiliary control files, and update files are
- File-naming conventions used in VM/XA SP service procedures
- How different control files can be used to create different systems.

The VM/XA System Product service process relies heavily on CMS files called control files. Each component of VM/XA System Product has at least one control file structure.

There are control files, auxiliary control files, and the actual update files (also called service files). These files may have the following generic filetypes:

Filetype	**File Contents**
CNTRL	Control file
AUX*xxxxx*	Auxiliary control file. *xxxxx* is up to five alphabetic or numeric characters.
UPDT*xxxx*	Update (listed in a CNTRL file; this is not recommended because it makes tracking difficult). *xxxx* is up to four alphabetic or numeric characters.
H*mmmmmxx*	Update (listed in an AUX file). *mmmmm* is an APAR number and *xx* is HP (for CP and dump viewing facility assembler source and object updates), PP (for other CP and dump viewing facility object updates), DS (for CMS modules), or CI (for GCS).
T*mmmmm*DS	CMS macro update (listed in an AUX file). *mmmmm* is an APAR number.

Control files may be extremely simple or quite elaborate, but their important function is to allow you to control those changes necessary to tailor your system to meet your current needs. Each structure of control files is a tree with the main control file at the base, auxiliary control files branching off from the main control file, and update files branching off from the auxiliary control files.

Figure 8 on page 378 shows the main CP control file, along with two auxiliary control files and their corresponding update files. (There are actually many auxiliary control files with update files). The control file HCPXA CNTRL contains several lines, one of which is the filetype of the VM/XA SP auxiliary control files (AUXXA). Every assemble file, macro, control block file, EXEC, or XEDIT macro, that has had service applied to it has a corresponding auxiliary control file called *fn* AUX*xxxxx*, where *fn* is the filename of the file that has been serviced. (For example, the auxiliary control file for program update service to HCPXXX ASSEMBLE is HCPXXX AUXXA.) Each auxiliary control file, in turn, contains the filetypes of the update files that have been applied.

Figure 8 on page 378 applies to all components of VM/XA SP. The structure for all components is identical, except that the main control filename and update file suffix vary (for example, the CMS main control file is called DMSXA CNTRL instead of HCPXA CNTRL; and the update file suffix is DS).

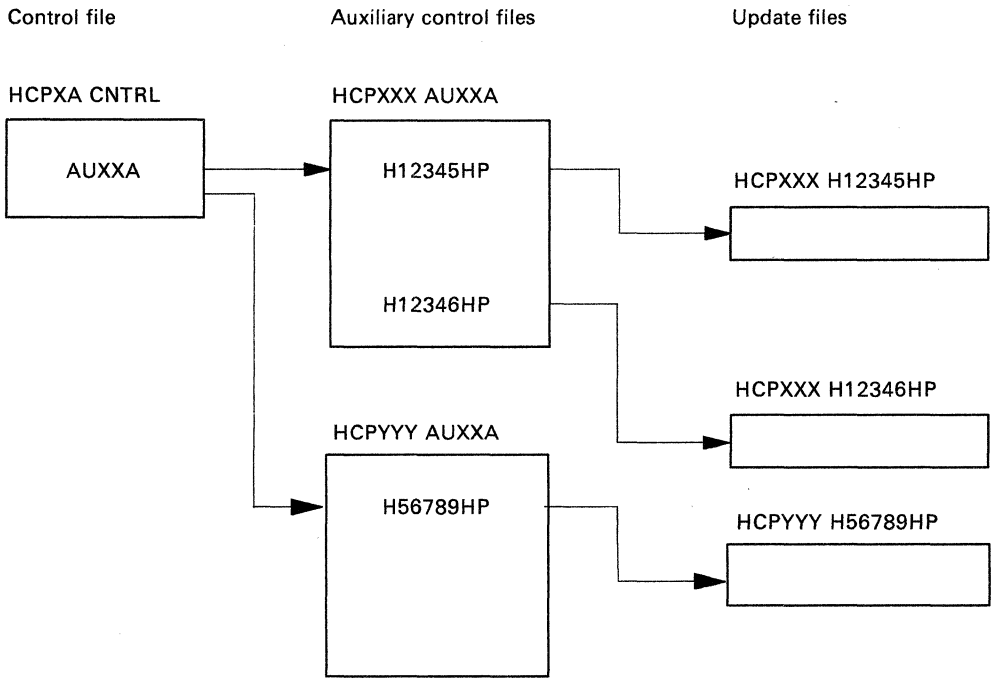


Figure 8. Service File Structure

Control Files

Control files are used in:

- Updating assembler source files and EXECs using XEDIT
- Updating control blocks and macros using XEDIT
- Creating and updating AUX files using VMFAPPLY
- Assembling source code using the VMFHASM EXEC procedure
- Patching object code using the PATCH facility
- Building MACLIBs using the VMFMAC EXEC procedure
- Regenerating updated EXECs using EXECUPDT
- Creating an executable CP or CMS nucleus using VMFBLD
- Generating CP utilities using UTILITY
- Generating CMS modules using CMSGEND.

Control files are used by the CMS UPDATE command. The EXECUPDT, XEDIT, VMFMAC, and VMFHASM procedures invoke UPDATE with the CTL option to modify source files. The VMFAPPLY, VMFBLD, and HCPLDR programs also use the control file structure directly; usually the same control file is used by all these procedures for a given product or component. For the purposes of product service, all the procedures require that the control file must have a filetype of CNTRL. The control filename is unique for each VM/SP XA component. The VM product control filenames and their contents are provided in "Main Control Files" on page 380.

Sample Control File

For an understanding of how the update procedures work, you should be familiar with the elements in a control file. A control file for the VM/XA System Product system might look like Figure 9 on page 379.

```
*THIS IS A SAMPLE CNTRL FILE FOR LOCAL CP UPDATES
*DO NOT USE THIS FILE
TEXT MACS HCPXA1 HCPXA2 CPLIB DMSSP CMSLIB OSMACRO
LOC1 AUXLCL LCL TX$ * COMMENT
TEXT AUXXA
```

Figure 9. Sample Control File

There are usually three kinds of records in a control file: MACS records, AUX file identification records, and comments. (A fourth kind, update file identification records, will work but should be avoided.)

MACS Records

A control file can have any number of MACS records. They must appear before any AUX file identification records. The first field in the MACS record is the update level identifier. You can think of this as a name field. If no update files are found, the filetype of the assembled text deck will be derived from the update level identifier on the MACS record: TEXT if the update level identifier is TEXT and TXT*xxxxx*, where *xxxxx* is the update level identifier, if it is anything else.

The remaining fields in the MACS record are the names of macro libraries. There may be up to eight macro library names on a single MACS record. VMFHASM uses the library list from the MACS record to issue a GLOBAL command before assembling the updated source file. The libraries are searched in the order specified.

AUX File Identification Records

A control file can have any number of AUX file identification (AUX*xxxxx*) records. The first field in an AUX file identification record is the update level identifier. If the auxiliary control file named in this record is found, but the top entry in the auxiliary control file contains no PTF number or local tracking number, the filetype of the assembled text deck will be derived from the update level identifier on the AUX file identification record: TEXT if the update level identifier is TEXT and TXT*xxxx*, where *xxxx* is the update level identifier, if it is anything else. (If more than one auxiliary control file is listed, the last one found—the one named on the top auxiliary control file identification record—determines the text deck filetype.)

The second field (sometimes called the update ID) is the filetype of the auxiliary control file. The characters AUX identify an auxiliary control file that lists additional fixes to be applied; AUXLCL in this example. AUXXA is the VM/XA System Product auxiliary control file, listing updates distributed by IBM. This file is listed at the bottom of the control file so that these updates are applied first. (Note that the file is read from the bottom to the top by UPDATE and from the top to the bottom by VMFBLD. UPDATE applies IBM changes first and local changes last, while VMFHASM names the text file, and VMFBLD finds it, according to the last change applied.)

Note: Service supplied by IBM for VM/XA System Product does not use preferred AUX files, but they are supported. If you wish to use them, see "Preferred AUX File" on page 665.

The third and fourth fields are optional. Both are 3-character fields. Either one may come first. One of them (for example, LCL in the third line of Figure 9) is the text deck filetype prefix. If the auxiliary control file corresponding to this record is the last file found, and if it contains a PTF number or appropriate local tracking number, the filetype of the assembled text deck will be *xxxnnnnn*, where *xxx* is the text deck filetype prefix and *nnnnn* is the five numeric digits (the last five characters) of the PTF number or local tracking number. (If no number is found, an error is signaled.) The letters LC in a text deck filetype prefix indicate a local fix. (All prefixes that do not begin with LC are reserved for IBM use.) The default text deck filetype prefix is TXT.

If the other 3-character field (usually placed after the text deck filetype prefix) is present, it contains the keyword TX$, indicating a patch.

Comments

A control file can have any number of comment records. Comment records are identified by an asterisk (*) in the first column. They may appear anywhere in the control file. Comments may also appear at the end of control records. They begin with an asterisk.

Update File Identification Records

For compatibility with other VM systems, VM/XA System Product also supports update file identification records in control files. Avoid using them if possible, because an update listed in a control file must be called UPDT*xxxx*. The four variable characters are not enough to identify the APAR with which this update is associated, essential information for tracking. List updates in an auxiliary control file, where the name can include an APAR number.

Varying Control Files to Generate Multiple Systems

By varying the control files, you can build different versions of your system from a single set of AUX and update files. For example, you might have the following files:

Filename	Filetype	Contents
PRODSYS	CNTRL	*THIS IS A SAMPLE CNTRL FILE *DO NOT USE THIS FILE TEXT MACS HCPXA1 HCPXA2 CPLIB DMSSP CMSLIB OSMACRO LC2 AUXLCL2 LCL TX$ P1 AUXP1 TEXT AUXXA
TESTSYS	CNTRL	*THIS IS A SAMPLE CNTRL FILE *DO NOT USE THIS FILE TEXT MACS **TESTLIB** HCPXA1 HCPXA2 CPLIB DMSSP CMSLIB OSMACRO **LC1 AUXLCL1 LCT TX$ * FREE FORM COMMENT** LC2 AUXLCL2 LCL TX$ P1 AUXP1 TEXT AUXXA

Notice that the control file for the test system contains one more level of updates (the record beginning with LC1) than the control file for the production system. When all the updates listed in the auxiliary control files have been applied, the system built with the test control file will contain more updated files than the system built with the production control file. Both control files serve as the base of a tree using the same set of AUX and update files, but the test tree has branches not in the production tree. It is not necessary to keep two sets of AUX and update files.

Notice also that the test system includes one more macro library (TESTLIB) than the production system.

Main Control Files

CP, CMS, CMS macros, the dump viewing facility, GCS, and the VM/XA SP loader each have a main control file. The contents of these main control files are shown below.

The contents of HCPXA CNTRL (control file for CP) are:

```
TEXT MACS  HCPXA1 HCPXA2 CPLIB DMSSP CMSLIB OSMACRO
PAT AUXPAT TX$    * Local Patches
LCL AUXLCL LCL    * Local Modifications
COR AUXCOR        * IBM Corrective Service
SUP AUXSUP        * SUP Reach Ahead
TEXT AUXXA        * IBM Put Service
```

The contents of DMSXA CNTRL (control file for CMS) are:

```
TEXT MACS DMSSP CMSLIB OSMACRO DOSMACRO CPLIB TSOMAC
PAT AUXPAT TX$    * Local Patches
LCL AUXLCL LCL    * Local Modifications
COR AUXCOR        * IBM Corrective Service
SUP AUXSUP        * SUP Reach Ahead
TEXT AUXXA        * IBM Put Service
```

The contents of DMSMXA CNTRL (control file for CMS macros) are:

```
TEXT MACS
PAT AUXMPAT TX$    * Local Patches
LCL AUXMLCL LCL    * Local Modifications
COR AUXMCOR        * IBM Corrective Service
SUP AUXMSUP        * SUP Reach Ahead
TEXT AUXMXA        * IBM Put Service
```

The contents of HCSXA CNTRL (control file for dump viewing facility) are:

```
TEXT MACS  HCPXA1 HCPXA2 CPLIB DMSSP CMSLIB OSMACRO
PAT AUXPAT TX$    * Local Patches
LCL AUXLCL LCL    * Local Modifications
COR AUXCOR        * IBM Corrective Service
SUP AUXSUP        * SUP Reach Ahead
TEXT AUXXA        * IBM Put Service
```

The contents of CSIXA CNTRL (control file for GCS) are:

```
TEXT MACS CSISP DMSSP CMSLIB
PAT AUXPAT TX$    * LOCAL PATCHES
LCL AUXLCL LCL    * LOCAL MODIFICATIONS
COR AUXCOR        * IBM CORRECTIVE SERVICE
SUP AUXSUP        * SUP REACH AHEAD
TEXT AUXXA        * IBM PUT SERVICE
```

The contents of HCPLDRCM CNTRL (control file for the VM/XA SP loader) are:

```
TEXT MACS HCPLDRM HCPXA1 HCPXA2 CPLIB DMSSP CMSLIB OSMACRO
TEXT AUXCMS
TEXT AUXXA
```

Notes:

1. HCPLDRCM CNTRL is used only to assemble the VM/XA SP loader.
2. All the service EXECs obtain the control filename from the product parameter file, except UTILITY EXEC, which always uses HCPXA CNTRL. HCPXA CNTRL is the usual control file for CP and for utilities.

Local Control Files

The control files supplied by IBM already provide for one level of local service; but when you create files for local service or updates of VM/XA System Product modules, you may still need to create a local control file. A local control file is a copy of the appropriate VM/XA System Product control file with an entry for each local auxiliary control file and the filename of your local MACLIB. For example, the file CPLCL CNTRL may contain:

```
TEXT MACS YOURMAC HCPXA1 HCPXA2 CPLIB DMSSP CMSLIB OSMACRO
LCT AUXLCT LCT  * TEST UPDATES
LCL AUXLCL LCL
TEXT AUXXA
```

Note: Control files (CNTRL) and auxiliary control files (AUX*xxxxx*) are read from the bottom up when updates are applied. The IBM-supplied auxiliary files should be the bottom entry in the control file so that during assembly the IBM updates are applied first.

Auxiliary Control Files

When auxiliary control files are distributed by IBM for VM/XA System Product, they have the filetype AUXXA. If an auxiliary control file is not distributed, it must be created by VMFAPPLY. Figure 10 shows a sample auxiliary control file for source updates. Figure 11 shows a sample auxiliary control file for patches to text decks.

Update ft (APAR #)	PUT Level	PTF #	Comments
H12567HP	801	UM98765	* COMMENT DESCRIBING FIX

Figure 10. Sample Auxiliary Control File for Source Updates

Update ft (Patch #)	Patch Indicator	APAR #	Comments
P23877	TX$	VM23877	* A PATCH TO BE REPLACED BY APAR VM23877

Figure 11. Sample Auxiliary Control File for Patches to Text Decks

For local source updates, a local tracking number takes the place of the PTF number in the auxiliary control file. Like the PTF number, a local tracking number has seven characters (two letters followed by five numbers).

When the text deck filetype is created, the two letters of the PTF number or local tracking number are ignored. The five numbers are appended to the text deck filetype prefix if one is present in the main control file. If no text deck filetype prefix is found, TXT is used as the prefix.

Update Files

Update files can contain:

- Updates to source code
- Patches to object code
- Prerequisite and corequisite information, without any patch or update.

All of the PTF and update files distributed by VM/XA System Product are assigned filetypes as follows:

Update File	Text Deck	APAR Number	PTF Number
H*mmmmm*HP H*mmmmm*PP H*mmmmm*DS H*mmmmm*CI T*mmmmm*DS	TXT*nnnnn*	VM*mmmmm*	UM*nnnnn*

where:

H indicates a VM/XA SP update.

T indicates a VM/XA SP CMS macro update.

HP is the 2-character identifier for VM/XA System Product CP and dump viewing facility assembler source and object updates.

PP is the 2-character identifier for other VM/XA System Product CP and dump viewing facility updates.

DS is the 2-character identifier for VM/XA System Product CMS.

CI is the 2-character identifier for VM/XA System Product GCS.

mmmmm is an APAR number. You can enter the command FILELIST * **mmmmm** * to see a listing of all the modules affected by APAR *mmmmm*. The filename is the same as the module name. Browse the AUX file for any one of these modules to find the corresponding PTF number.

nnnnn is a PTF number. You can enter the command FILELIST * **nnnnn** * to see a listing of all the modules affected by PTF *nnnnn*. The filename is the same as the module name. Browse the AUX file for any one of these modules to find the corresponding APAR number.

For example, the code and fixes to answer APAR VM12567 against the CP module HCPCFM in VM/XA System Product are contained in the file HCPCFM H12567HP.

The file HCPCFM AUXXA contains the entry:

```
H12567DK    801       UM98765  *COMMENT DESCRIBING FIX
```

UM98765 is the PTF corresponding to APAR VM12567. If UM98765 is the last PTF applied to HCPCFM, the assembled text deck will be called HCPCFM TXT98765.

For your local updates, assign a local tracking number with the format LC*nnnnn*, where the prefix LC indicates a local fix. The five numeric digits of the local tracking number can be used to name text decks in the same way as the five numeric digits of the PTF number.

Guidelines for Using Update Files

1. Keep each fix (patch or APAR) in a separate update file. This applies to both source update and patch fixes. You may have several files containing the same fix (this happens when the same fix applies to several parts), but you must never have several fixes in the same file.

   Each fix should have a unique identifier for control purposes. This identifier is the filetype of the update file. If the same fix applies to several parts, there should be a file for each part, all with the same filetype.

2. Keep all local fix descriptions for the same part in the same AUX file, unless a fix applies to a different control file level.

   Local fixes for the same part should not be distributed over AUX files (different control file levels) arbitrarily. Local service should be easily distinguished from IBM service and should always be applied last. Local service can be distributed over separate control files for the purpose of maintaining different service levels with a single structure of AUX and update files. Each level can be built from a different control file containing only the desired level identifiers.

3. Never place local patches in AUX files from IBM. In other words, keep your local service separate from IBM service. Local service should be easily distinguished from IBM service and should always be applied last.

4. Patches to text files should be applied only when no source file is available. If you do apply text file patches when source code is available, they should be converted to source updates and reassembled before they are moved from a test environment to a production-level system.

   Building local source updates on top of local text file patches for the same part will lead to confusion.

5. Do not place the names of update files directly in the main control file. Place update filenames in an auxiliary control file.

   If you place the update filename in the main control file instead of an auxiliary control file, the update file's filetype must be UPDT*xxxx*. If you place it in an auxiliary control file, the update file's filetype can be derived from the APAR number, and you can specify the text deck's filetype (derived from the PTF number). These filetypes are used during assembly and nucleus build.

Chapter 8. Files Used in Program Update Service and Corrective Service

This chapter describes:

- The files used in applying program update service and corrective service
- The program update tape (PUT)
- The corrective service tape.

Files Used in Program Update Service and Corrective Service

Besides the source or object code to be updated, the following files are used to apply program update service and corrective service:

- PUT document
- COR document
- PUT descriptor file
- COR descriptor file
- Product contents directory
- Memo to Users
- Control file (see Chapter 7, "How VM/XA System Product Uses Control Files and Update Files" on page 377)
- Auxiliary control files (see Chapter 7, "How VM/XA System Product Uses Control Files and Update Files" on page 377)
- Update files, also called service files
- Update shells
- Text decks
- Text shells
- PTF parts list
- Apply list
- Exclude list
- Exception log
- Receive history file
- Load list
- Temporary load list
- Load map
- Restart indicator files
- $LEVEL MAP
- $LEVEL EXEC
- $LEVEL $TAPE
- Product parameter file
- Product parameter override file
- Temporary product parameter file
- Service disk map.

The PUT Document

The PUT document describes the service procedure contained on the program update tape. It lists the steps involved in applying service to the system, and contains a description of the VMSERV EXEC. (Other service installation EXECs are described in the Memo to Users.) Read the PUT document before you try to apply service.

The COR Document

The COR document describes the service procedure contained on the corrective service tape. It lists the steps involved in applying service to the system, and contains an explanation of the service EXECs. Read the COR document before you try to apply service.

The PUT Descriptor File

The PUT descriptor file is a directory of the products and components for which service is contained on the program update tape. It shows how many files are available for each product and component and where they are. The PUT descriptor file is called PUT *nnnn*, where *nnnn* is the PUT number. Figure 12 is an example of a PUT description file.

```
Record One
    VM System Program Update Tape
Record Two
    VOLnn of 03
Record Three to n     *** Multi-volume directory
    :VOL01.
    PUT   FILES     02
    prodid1 HDR     01
    prodid1 comp1   nn
    prodid1 comp2   nn
    prodid1 comp3   nn
    prodid2 HDR     01
    prodid2 comp1   nn
    prodid2 comp2   nn
    :VOL02.
    PUT   FILES     02
    prodid2 HDR     01
    prodid2 comp3   nn
    prodid2 comp4   nn
    prodid3 HDR     01
    prodid3 comp1   nn
    prodid3 comp2   nn
    :VOL03.
    PUT   FILES     02
    prodid3 HDR     01
    prodid3 comp2   nn
    prodid3 comp3   nn
```

Figure 12. Example of a PUT Descriptor File

The first record of the PUT descriptor file states that it applies to a program update tape. (This record distinguishes the PUT descriptor file from the COR descriptor file, described in "The COR Descriptor File" on page 387, which is very similar.) The second record tells how many volumes the tape has and which volume this is. If there is more than one volume, the PUT descriptor file appears on each volume, but describes the whole tape.

The other records describe the contents of the file. A :VOL*nn* record indicates the beginning of each volume. Each of the three-word records following a :VOL*nn*. record describes a group of tape files. (A tape file is the data between two tape marks. It may contain more than one CMS file.) If the first two words are PUT FILES, that group of files is the two standard tape files that begin every volume of the program update tape: one containing the PUT descriptor file, PUT document, and service EXECs, the other containing informational files about all the products for which service is supplied on the tape (see Figure 33 on page 430). Otherwise, the first word names the product to which the files apply; the second word is either HDR (meaning a header file) or the name of the component to which the files apply. The third word is always the number of files in the group.

Note: A product may be entirely on a single volume (*prodid1*); that it may cross volumes at a component break (*prodid2*); or that a single component may span multiple volumes (*prodid3 comp2*). In the case of a product (or component) crossing a volume boundary, the break will be indicated in the product contents directory by the inclusion of a :VOL*nn* record, where *nn* indicates the continuation volume.

The COR Descriptor File

The COR descriptor file is a directory of the products and components for which service is contained on the corrective service tape. It shows how many files are available for each product and component and where they are. The COR descriptor file is called COR *ymdd*, where *y* is the last digit of the year, *m* is the month (numbered 1 through C in hexadecimal), and *dd* is the day; for example, 8B05 is November 5, 1988. The COR descriptor file has the same format as the PUT descriptor file (see "The PUT Descriptor File" on page 386). Figure 13 is an example of a COR descriptor file.

```
VM Corrective Service Tape 9110 1X660 PSIP SP2NP31 89/01/10 14:02:04.157208
VOL01 of 01
:VOL01.
COR      FILES    02
56643082 HDR      01
56643082 CP       08
56643082 CMS      09
56643082 DV       07
56643082 GCS      03
```

Figure 13. Example of a COR Descriptor File - COR 9110

The Product Contents Directory

There is a product contents directory for each new format product serviced on the program update tape or corrective service tape. The product contents directory lists the tape files supplied for each component of each product, and must be repeated if the product crosses a tape boundary. The product contents directory can be seen as an expansion of the PUT descriptor file or COR descriptor file: the descriptor file tells you how many files you have and where they are; the product contents directory tells you what the files listed in the descriptor file are. Figure 14 on page 388 is an example of a product contents directory.

The filename of the product contents directory is the product ID. The filetype is:

- On the program update tape: $PUT*nnnn*, where *nnnn* is the PUT number
- On the corrective service tape: $COR*ymdd*, where *y* is the last digit of the year, *m* is the month (numbered 1 through C in hexadecimal), and *dd* is the day; for example, 8B05 is November 5, 1988.

prodid1 **$PUT***nnnn*

```
:VOL01.
:CP.
AXLIST
PARTLST
UPDT
TEXT
MACAUX
MACUPDT
SOURCE
MACLIB
EXEC
RESV

:CMS.
AXLIST
PARTLST
UPDT
TEXT
MACAUX
MACRO
MACUPDT
SOURCE

:VOL02.
MACLIB
EXEC
MODULE
HELP
IOCP
RESV

:DV.
AXLIST
PARTLST
DVFTEXT
DVFMODS
RESV
```

prodid1 **$COR***ymdd*

```
:VOL01
:CP.
AXLIST
PARTLST
UPDT
TEXT
MACAUX
MACUPDT
MACRO



:CMS
AXLIST
PARTLST
UPDT
TEXT
MODULE
```

Figure 14. Examples of Product Contents Directories

The Memo to Users - 56643082 MEMO

You'll receive a Memo to Users for each licensed program that needs service. The Memo contains instructions for servicing each program, and describes the service installation EXEC (if any) for each program. You should read the Memo for each licensed program before you try to apply service.

The Memo To Users for VM/XA System Product Release 2 is called 56643082 MEMO.

Update Files

There are two types of update files:

- Source update files for fixes to source code
- Patch update files for fixes to object code.

There is one update file for each part affected by an APAR.

Source Update Files

Source update files are named *filename XnnnnnYY* where

- *filename* is the name of the part being serviced
- *nnnnn* is the APAR number
- *X/YY* is the value assigned to the :SLVI. tag in the product parameter file.

Each source update file contains:

- Requisite information, consisting of PREREQ, CO-REQ, and IF-REQ entries
- DEPEND entries added by VMFAPPLY, identifying any PTFs for which this PTF is a requisite
- Update control statements and assembler language statements defining an effective change to a module's source code. (These changes are actually made as object code is created.)

Source update files are shipped on the PUT or corrective service tape. When a source update is applied, VMFAPPLY changes the filemode of the update file from 1 to 5. Figure 15 is an example of a source update file.

```
./ * PREREQ: VM33332 VM31452
./ * CO-REQ: NONE
./ * IF-REQ: NONE
./ * DEPEND: H33330DS H34594DS
./ I 09180000          $ 9182000 2000          08/16/88 20:56:22
         TM    MISFLAG2,OSTYPLD WAS THIS AN OS TYPE LOAD?      @VA33331
*                           (WERE WE CALLED BY DMSSLN?)       @VA33331
         BO    LDRB0231       Don't reset EPA to new value     @VA33331
./ I 09190000          $ 9190500 500           08/16/88 20:56:22
LDRB0231 DS    0H                                             @VA33331
```

Figure 15. Example of a Source Update File - DMSLDR H33331DS

Patch Update Files

The customer creates patch update files for parts of the CP nucleus, CMS nucleus, and GCS nucleus serviced by replacement; that is, parts for which no source code is available. Patch update files contain:

- Requisite information, consisting of PREREQ, CO-REQ, and optional IF-REQ entries
- VERIFY and REPLACE statements that effectively change the object code as the nucleus is created. When you apply a patch update, change the filemode of the update file from 1 to 5. Figure 16 on page 390 is an example of a patch update file.

```
./ * * PREREQ: NONE
./ * * CO-REQ: NONE
./ * NAME HCPAFF
./ * * This is a patch.  It
./ * *   will be replaced by
./ * *   APAR VM23877.
./ * VER 61C 0780
./ * REP 61C 0700 ■
```

Figure 16. A Sample Patch Update File

Update Shells

VMFREC creates update shells on the APPLY disk for parts on replacement service. Update shells use the same naming convention as update files. An update shell takes the place of the source update file that would exist if the part were on update service. Update shells contain requisite information and DEPEND entries (created by VMFAPPLY), but no executable code. Like source update files, update shells have the module name for a filename and a filetype derived from the APAR number. When the PTF associated with the APAR is applied, VMFAPPLY changes the filemode of the update shell from 1 to 5. Figure 17 is an example of an update file shell.

```
./ * PREREQ: NONE
./ * CO-REQ: NONE
./ * IF-REQ: NONE
./ * * THIS IS AN UPDATE FILE SHELL
```

Figure 17. Example of an Update File Shell - DMSMGC H34362DS

Text Decks

The self-documenting text decks for CP and CMS service contain a prolog with the following information:

1. A description of the APAR, including the APAR number, PUT level, PTF number, and content (from the AUX file)
2. Prerequisites and corequisites for this service, consisting of PREREQ, CO-REQ, and IF-REQ entries and agreeing with the requisites in the update file and update shell
3. A date and time stamp
4. The macro libraries used (optional).

The text decks also contain assembler text consisting of ESD, TXT, RLD, and END cards.

If the filetype of an update file listed in a text deck begins with a percent sign (%), that update is listed in the control file and not in an auxiliary control file. Figure 18 on page 391 is an example of a text deck.

```
H33398DS 203 UM90033 NLS FILE NAMING CONVENTIONS ENHANCEMENT
*         DMSMGC   H33398DS C1 XA1396  09/13/88   14:34:32
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
H34362DS 203 UM04083 GENMSG CALCULATES INDEX PAST END OF PAGE.
*         DMSMGC   H34362DS A5 XA1191  11/03/88   13:03:06
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
*         DMSSP    MACLIB   S2 XA1390  10/21/88   10:43:25
  ⋮
*         DMSMGC   DMSTSP1  A1 XA1191  11/03/88   13:10:22
¢ESD ...
¢TXT ...
  ⋮
¢TXT ...
¢RLD ...
¢END ...
```

Figure 18. Example of a Text Deck - DMSMGC TXT04083

Text Shells

VMFREC creates a text shell on the DELTA disk for each PTF listed in the prolog of a text deck for which neither a text deck nor a text shell is available on the PUT or on the DELTA or LOCAL minidisks. The text shell contains:

1. A description of the APAR, including the APAR number, PUT level, PTF number, and content (from the AUX file)
2. Prerequisites and corequisites for this service, consisting of PREREQ, CO-REQ, and IF-REQ entries and agreeing with the requisites in the update file and update shell
3. A date and time stamp
4. The macro libraries used (optional).

The text shell does not include the executable text. VMFBLD will not accept a text shell as input. The shell is provided so that a search by PTF number will identify all the modules affected by the PTF.

If the filetype of an update file listed in a text shell begins with a percent sign (%), that update is listed in the control file and not in an auxiliary control file. Figure 19 is an example of a text shell.

```
H33398DS 203 UM90033 NLS FILE NAMING CONVENTIONS ENHANCEMENT
*         DMSMGC   H33398DS C1 XA1396  09/13/88   14:34:32
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
H34362DS 203 UM04083 GENMSG CALCULATES INDEX PAST END OF PAGE.
*         DMSMGC   H34362DS A5 XA1191  11/03/88   13:03:06
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
```

Figure 19. Example of a Text Shell - DMSMGC TXT04083

The PTF Parts List

The filename of the PTF (program temporary fix) parts list is the PTF number and the filetype is $PTFPART. It contains a list of the parts needed to install the PTF. This list should identify the appropriate text file even when the text file is not shipped on the PUT. The filemode of the PTF parts list is changed to filemode 5 after the PTF is applied.

The Apply List

The apply list contains a list of the PTFs to be applied to a product. The first line in the apply list is always a comment identifying the product and PUT level. An asterisk (*) indicates a comment and may be used to exclude a PTF. Figure 20 is an example of an apply list.

```
* APPLY LIST FOR PRODUCT 56643082 COR 9110
UM03532
UM03605
UM03603
UM03601
UM03586
UM03610
UM03590
UM03693
UM03728
UM03782
UM03790
UM03794
UM03827
UM03993
UM90033
UM04083
UM04165
UM04462
```

Figure 20. Example of an Apply List - DMSXA $APPLIST

The PUT contains an apply list for each PUT level and one for all the levels on the PUT tape. The filename of the apply list on the service tape is the same as the filename of the control file specified in the product parameter file. (If you change the filename of the control file in the PPF, the filename of the apply list is automatically changed too.) The filetype is:

- For program update service:
  - $APP*nnnn* for the apply list associated with a single PUT level, where *nnnn* is the PUT level
  - $APPALL for the cumulative apply list.
- For corrective service:
  - $APC*ymdd* for the apply list associated with a single corrective service tape, where *ymdd* is the date (last digit of the year, month numbered in hexadecimal, day)
  - $APCALL for the cumulative apply list. (Since corrective service tapes are not cumulative, this is the same as the apply list associated with a single COR tape. It is included for consistency.)

VMFREC will copy the cumulative apply list to a file whose filetype is $APPLIST, and append any cumulative apply list it finds on the intermediate alternate disk(s). *fn* $APPLIST is the apply list used by the VMFAPPLY EXEC. If you want VMFAPPLY to use a different apply list, you must copy it with a filetype

of $APPLIST. (Use the REPLACE option.) Copy it instead of renaming it so that you are sure to keep the apply list you used.

The Exclude List

The exclude list contains a list of the PTFs that are not to be applied to a product, even though they are listed in the apply list. Entries must be in the same order as in the apply list. The first line in the exclude list is always a comment identifying the product and PUT level. Figure 21 is an example of an exclude list.

```
* EXCLUDE LIST FOR PRODUCT 56643082 COR 9110
UM03601
UM90033
```

Figure 21. Example of an Exclude List - DMSXA $EXCLIST

The PUT contains an exclude list for each PUT level and one for all the levels on the PUT tape. The filename of the exclude list on the service tape is the same as the filename of the control file and apply list. The filetype is:

- For program update service:
  - $EXP*nnnn* for the exclude list associated with a single PUT level, where *nnnn* is the PUT level
  - $EXPALL for the cumulative exclude list.
- For corrective service:
  - $EXC*ymdd* for the exclude list associated with a single corrective service tape, where *ymdd* is the date (last digit of the year, month numbered in hexadecimal, day)
  - $EXCALL for the cumulative exclude list. (Since corrective service tapes are not cumulative, this is the same as the exclude list associated with a single COR tape. It is included for consistency.)

VMFREC will copy the cumulative exclude list to a file whose filetype is $EXCLIST, and append any cumulative exclude list it finds on the intermediate alternate disk(s). *fn* $EXCLIST is the exclude list used by the VMFAPPLY EXEC. If you want VMFAPPLY to use a different exclude list, you must copy it with a filetype of $EXCLIST. (Use the REPLACE option.) Copy it instead of renaming it so that you are sure to keep the exclude list you used.

Note: If two PTFs are in the same text deck, you cannot exclude the first unless you also exclude the second, even if you list the first one on the exclude list. This restriction applies because the first PTF is considered to be a prereqisite for the second PTF.

If you list the first PTF in the exclude list, but not the second, and if the second PTF appears in the apply list, both PTFs are applied. You receive an error message indicating that a PTF you wanted to exclude has been applied.

The Exception Log

The exception log is called $VMF*XXX* $ERRLOG, where *XXX* can be:

- REC for the receive exception log created by the VMFREC EXEC
- APP for the apply exception log created by the VMFAPPLY EXEC
- BLD for the build exception log created by the VMFBLD EXEC.

The exception log contains any error or informational messages issued by its corresponding exec and is written to the A-disk. If the exception log already exists when the exec is invoked, a date and time stamp is inserted to separate the earlier log entries from the new entries. You should browse the exception log with

the VMFVIEW EXEC after VMFREC, VMFAPPLY, or VMFBLD finishes processing. Table 11 on page 394 shows the different types of messages that may appear in an exception log.

Table 11. Exception Log Message Codes

Code	Explanantion
ST:	A status message pertaining to the major function of the current process
WN:	A warning message
CK:	A message that **MUST** be checked
SV:	A severe problem encountered
MS:	Mismatched parts, such as AUX files and AUX entries in the front of text decks
RQ:	Messages pertaining to requisites
RO:	Messages pertaining to requisites outside the component
BD:	Informational messages issued during the build process

Note: All messages except status messages **must** be investigated. A warning message does not necessarily mean that anything is wrong, but **you cannot be sure until you check it out.**

Exception logs are cumulative. The most recent entries are at the **top**.

The Receive Exception Log

```
*****************************************************************
*****    RECEIVE  -  04/13/89     -     13:29:10  - run     *****
*****************************************************************
*****     Product:  56643082      Component:  CMS           *****
*****************************************************************
```

Figure 22 (Part 1 of 2). Example of a Receive Exception Log - $VMFREC $ERRLOG

```
ST:DMSREC1852I This is VOL01 of 01, level 9110 COR tape.
ST:DMSREC1806I The current SERVICE DISKMAP file contains
ST:            the map of the mounted tape.
ST:DMSREC1804I Receiving service for component CMS of product 56643082
ST:DMSREC1940I Merge processing is not required for component CMS.
ST:DMSREC1851I Processing AXLIST  with the part handler VMFRCAXL EXEC.
ST:DMSREC1851I Processing PARTLST  with the part handler VMFRCCOM EXEC.
ST:DMSREC1851I Processing UPDT  with the part handler VMFRCUPD EXEC.
ST:DMSREC1851I Processing TEXT  with the part handler VMFRCTXT EXEC.
ST:DMSREC1851I Processing MACAUX  with the part handler VMFRCAUX EXEC.
ST:DMSREC1851I Processing MACUPDT  with the part handler VMFRCUPD EXEC.
ST:DMSREC1851I Processing EXEC  with the part handler VMFRCCOM EXEC.
ST:DMSREC1851I Processing MODULE  with the part handler VMFRCCOM EXEC.
ST:DMSREC1851I Processing HELP  with the part handler VMFRCCOM EXEC.
ST:DMSREC1851I Processing HELP  with the part handler VMFRCUPP EXEC.
ST:DMSREC1850I The processing for RESV by the VMFRCCOM EXEC was bypassed.
        ****************************************************************
        *****    RECEIVE  -  04/13/89     -     13:28:58  - run     *****
        ****************************************************************
        *****                INFO option                              *****
        ****************************************************************
ST:DMSREC1852I This is VOL01 of 01, level 9110 COR tape.
```

Figure 22 (Part 2 of 2). Example of a Receive Exception Log - $VMFREC $ERRLOG

The Apply Exception Log

```
        ****************************************************************
        *****    APPLY    -  04/13/89     -     13:42:26  - run     *****
        ****************************************************************
        *****     Product:  56643082      Component:  CMS           *****
        ****************************************************************
MNT191  191   A    R/W
MNT395  395   B    R/W
MNT691  691   E    R/O
MNT391  391   F    R/O
MNT692  692   G    R/O
MNT392  392   H    R/O
MNT593  593   I    R/O
MNT293  293   J    R/O
MNT193  193   K    R/O
MNT393  393   L    R/O
```

Figure 23 (Part 1 of 2). Example of an Apply Exception Log - $VMFAPP $ERRLOG

```
MNT190  190   S    R/O
MNT19E  19E   Y/S  R/O
ST:DMSAPP1853I Processing PTF UM03532
ST:DMSAPP1851I Processing DMSLDR TXT03532 with the part handler VMFAPTXT EXEC.
WN:DMSWTL1885W ***************************************************
WN:            ***  DMSLDR   AUXLCL   listed in DMSXA    CNTRL    ***
WN:            ***  represents service at a higher level than     ***
WN:            ***  DMSLDR   AUXCOR  .  The higher level service  ***
WN:            ***  may need to be reworked or removed.           ***
WN:            ***-----------------------------------------------***
WN:            ***  DMSLDR   AUXLCL                               ***
WN:            ***-----------------------------------------------***
WN:            ***  L00003LC LCL LCL00003                         ***
WN:            ***  L00002LC LCL LCL00002                         ***
WN:            ***  L00001LC LCL LCL00001                         ***
WN:            ***************************************************
BD:DMSAPX1858I CMSLOAD build list contains files that
BD:            have been serviced.
ST:DMSAPP1851I Processing DMSLDR H33150DS with the part handler VMFAPCOM EXEC.
ST:DMSAPP1853I Processing PTF UM03605
ST:DMSAPP1851I Processing DMSLDR TXT03605 with the part handler VMFAPTXT EXEC.
ST:DMSAPP1851I Processing DMSLDR H33331DS with the part handler VMFAPCOM EXEC.
ST:DMSAPP1853I Processing PTF UM03603
ST:DMSAPP1851I Processing DMSLDR TXT03603 with the part handler VMFAPTXT EXEC.
ST:DMSAPP1851I Processing DMSLDR H33330DS with the part handler VMFAPCOM EXEC.
ST:DMSAPP1856I PTF UM03601 is in the DMSXA $EXCLIST
ST:            exclude list for product ID 56643082, and
ST:            will not be applied.
  :
WN:DMSAPP1823W The UM03844 $PTFPART PTF parts list file was not found.
WN:            A PTF might be missing for product ID 56643082.
WN:DMSAPP1823W The UM03894 $PTFPART PTF parts list file was not found.
WN:            A PTF might be missing for product ID 56643082.
```

Figure 23 (Part 2 of 2). Example of an Apply Exception Log - $VMFAPP $ERRLOG

The Build Exception Log

```
       ****************************************************************
       *****    BUILD    -  04/13/89     -     14:30:09  - run     *****
       ****************************************************************
       *****     Product:  56643082      Component:  CMS           *****
       ****************************************************************
ST:DMSBLD1851I Processing CMSLOAD  with the part handler VMFBDNUC EXEC.
RQ:DMSBNC1857I APAR VM30394 is a required requisite, but
RQ:            part DMSSCRCB T30394DS on disk 293 has not
RQ:            been applied by VMFAPPLY and the update file
RQ:            is not fm5.
  :
```

Figure 24. Example of a Build Exception Log - $VMFBLD $ERRLOG

The Receive History Log

The receive history log has a filename of $VMFREC and a filetype of $HISTORY. The receive history log contains a list of all CMS files received from the PUT when VMFPLC2 is invoked by VMFREC for any component of VM/XA SP, and shows the date and time when each file is received.

Note: Files loaded from tape using any of the INFO options of VMFREC EXEC do not appear in the receive history log. Therefore, the various tape documents, the service disk map, and the service EXECs will not appear in the receive history log.

The receive history log is cumulative. It differs from the exception logs in that the most recent entries are at the **bottom**.

```
**********************************************************
Receiving: 56643082 - CMS 13 Apr 1989 13:29:31
**********************************************************
 LOADING.....
 DMSXA    $APCALL  B1
 DMSXA    $APC9110 B1
 DMSXA    $EXCALL  B1
 DMSXA    $EXC9110 B1
 END-OF-FILE OR END-OF-TAPE
 LOADING.....
 UM03532  $PTFPART B1

  :
**********************************************************
End Receive for: 56643082 - CMS 13 Apr 1989 13:31:00
**********************************************************
```

Figure 25. Example of a VMFREC History File - $VMFREC $HISTORY

The Load List

The load list is an ordered listing of the CP, CMS, or GCS modules. The ordering (top to bottom) determines which modules come first on the system residence device and which modules come into storage upon an initial program load.

The CP Load List

For the CP load list, the order is determined by the HCPMDLAT MACRO, which is invoked by the HCPLDL ASSEMBLE module. The load list is created when HCPLDL is assembled. CPLOAD EXEC is shipped on the PUT service tape, but is not shipped on the Corrective Service tape. Therefore UTILITY must be used to generate the CPLOAD EXEC only if corrective service is applied or if local modifications are made.

The file created by assembling HCPLDL is called HCPLDL TEXT or HCPLDL TXT*xxxxx*. If it is called HCPLDL TXT*nnnnn*, you must rename it HCPLDL TEXT. You must then use the UTILITY EXEC to convert HCPLDL TEXT to CPLOAD EXEC.

The modules fall into these general categories:

- Resident modules. These modules are nonpageable.
- Pageable modules loaded at initialization. These modules may not be paged until system initialization has finished. Thereafter, the modules can be paged out.

- Other pageable modules. These modules, along with the modules that are resident only during initialization, comprise the part of the system that CP pages in and out of real storage.

The CP load list must have fixed-length 80-byte records.

Do not change the CP load list unless it is absolutely necessary.

The CMS Load List

The CMS load list is called CMSLOAD EXEC. Do not change anything in the CMS load list except the SLC (Set Location Counter) cards, which point to files that determine where the modules in the CMS resident nucleus are loaded.

The GCS Load List

The GCS load list is called GCSLOAD EXEC. Do not change anything in the GCS load list.

The Temporary Load List

The temporary load list is a file generated by VMFBLD. It includes all of the text decks in the original load list (CPLOAD EXEC for CP, CMSLOAD EXEC for CMS, GCSLOAD EXEC for GCS, etc.) as well as the filetype of the "latest" version of the text deck (latest being determined by the CNTRL and AUX file structure). HCPLDR MODULE generates the CP, CMS, and GCS nuclei using the temporary load list for each component.

```
&TRACE OFF
&1 &2 &3 HCPLDR LOADER
&1 &2 &3 HCPSYS TXT12345
&1 &2 &3 HCPRIO TXT22333
&1 &2 &3 HCPDDI TXT98765
  :
&1 &2 &3 HCPMM3 TXT22333
```

Figure 26. Example of a Temporary Load List - $$$TLL$ EXEC

The Load Map

The load map has the same filename as the control file and a filetype of LOADMAP. It contains the map of CSECT external symbol resolution and service level information from the text decks, which are self-documenting. It is written to the A-disk if you specify the DISK option for the VMFBLD EXEC. Otherwise, it is spooled to the printer.

If the filetype of an update file listed in the load map begins with a percent sign (%), that update is listed in the control file and not in an auxiliary control file.

Restart Indicator Files

The VMFREC EXEC stores one or two restart indicator files on the first target disk in the string used during the receive process. Do not erase the restart indicator files. If you need to restart VMFREC, they determine how VMFREC will proceed.

The restart indicator files are called:

- For program update service:

  prodid $MRP*nnnn*
  prodid $PUT*nnnn*

  where:

  prodid
  is the identifier of the product for which service is being received, for example, 56643082.

  nnnn
  is the PUT level.

- For corrective service:

  prodid $MRC*ymdd*
  prodid $COR*ymdd*

  where:

  prodid
  is the identifier of the product for which service is being received, for example, 56643082.

  ymdd
  is the date (last digit of year, month numbered in hexadecimal, day).

If the $MR*xxxxx* file is present, the files for PUT level *nnnn* or corrective service dated *ymdd* have been received and VMFREC has successfully completed. The MERGE disks will be merged from alternate to current when VMFREC is rerun for the same product or component. The components for which service have been received are listed in the $MR*xxxxx* file.

prodid $PUT*nnnn* or *prodid* $COR*ymdd* contains the first record of the PUT or COR descriptor file, plus the date when VMFREC was run and, for corrective service, the time. If this file is present for a tape you want to receive, it indicates that VMFREC has already been run for this tape.

The $LEVEL MAP File

The $LEVEL MAP file is the inventory of service tool parts that may be in files 1 and 2 of the service tape. It lists all the parts of a tested level of the service EXECs that exist in file 1 of the tape as well as the total set of $PPF files that can possibly be found in file 2 of the tape. The format is the following:

- Line 1: The fifth word is the level of the service EXECs in file 1. (The asterisk at the start of the line counts as one word.)

  Only the fifth word is significant.

- Line 2: Comment heading of file types for XA/SP2 and 370/SP.

  The columns are:

  - file name
  - real file type
  - PTF-numbered file type for XA/SP2
  - PTF-numbered file type for 370/SP.

  The service EXEC parts distributed in file 1 are XA/SP2 parts which for a given PTF-numbered level are functionally equivalent to the 370/SP 6 PTF-numbered parts listed in column 4.

- Lines 3-n: The list of executable service parts.
- Line n+1: End of executable parts indicator.
- Lines n+2-n: List of $PPF files that exist.

This list contains the names of all existing product $PPF files. The service tape contains some subset of these $PPF files depending upon the product set it contains. If an ordered product has a $PPF associated with it, that $PPF will be on the tape.

```
* SES PUT LEVEL 8901
*                XA/SP2  370/SP6
VMFREC   EXEC   EXC12345 EXC12346
VMFRCTXT EXEC   EXC12345 EXC12346
   .
   .
   .
$VMFMSG$ EXEC   EXC23123 EXC23124
VMFLDS   MODULE MOD12345 MOD12346
* PPFs in the second tape file
56643082 $PPF   $PF12345
   .
   .
   .
5664167E $PPF   $PF12346
```

Figure 27. Example of a $LEVEL MAP file

The $LEVEL EXEC File

The $LEVEL EXEC file is the inventory of service EXEC parts that are installed on your system. It has the same format as the $LEVEL MAP file but its content is slightly different. It reflects the union of the service EXEC parts and $PPF files that the customer has actually received using either VMFREC INFO or VMFREC prodid.

When receiving service EXEC parts using VMFREC prodid, you must manually update the $LEVEL EXEC file to reflect the installed level of the service EXECs. To do this, update the level word in line 1 with a suffix (for example 8901A) and update the body of $LEVEL EXEC to reflect the new PTF-numbered file type for the part. If you run "VMFREC INFO (EXECS", it creates or updates the $LEVEL EXEC file to reflect the service EXEC parts loaded from the tape.

The $LEVEL $TAPE File

This file is a copy of the tape map for the first two files (the header files) of the service tape. It may be used to check the contents of the tape against the $LEVEL MAP tape inventory file if discrepancies are reported. Figure 28 on page 401 shows a sample $LEVEL $TAPE file.

```
SCANNING....
COR      8B18     A1
COR      DOCUMENT A1
$$LDR$$  XED12348 A1
$VMFCHK$ XED12347 A1
$VMFMSG$ EXC12345 A1
$VMFPAT$ EXC12346 A1
VMFAPCOM EXC12345 A1
VMFAPNLS EXC12349 A1
VMFAPPLY EXC12348 A1
VMFAPTXT EXC12347 A1
VMFASM   EXC12346 A1
VMFBDCPY EXC12345 A1
VMFBDNUC EXC12349 A1
VMFBLD   EXC12348 A1
VMFHASM  EXC12347 A1
VMFLDS   MOD12346 A1
VMFOVER  EXC12349 A1
VMFRCAXL EXC12346 A1
VMFRCCOM EXC12345 A1
VMFRCTXT EXC12349 A1
VMFRCUPD EXC12348 A1
VMFRCUPP EXC12347 A1
VMFREC   EXC12346 A1
VMFSETUP EXC12345 A1
$LEVEL   MAP      A1
END-OF-FILE OR END-OF-TAPE
56643082 $PF12345 A1
56643082 MEMO     A1
56643082 0200241  A1
END-OF-FILE OR END-OF-TAPE
```

Figure 28. Example of a $LEVEL $TAPE File

The Product Parameter File

You must have a product parameter file in order to update products supported by the new service EXECs, including VM/XA System Product. The filename of the product parameter file is the product ID—for example, 56643082—and the filetype is $PPF.

The product parameter file consists of a product area, one or more component areas, and one or more override areas.

- The product area (introduced by the :COMPLST. tag) lists the components of the product.
- Each component area (introduced by a *:compname.* tag) has three sections:
  - The first section (introduced by the :CNTRLOP. tag) contains control options such as the control file name, the system level and version, and the national language.
  - The second section (introduced by the :MDA. tag) contains minidisk assignments:
    - The TASK disks, where you can isolate the latest level of service EXECs
    - The LOCAL disks, where you keep local update files
    - The APPLY disk, where the auxiliary control structure will be built
    - The DELTA disks, where you will load the update files from the program update tape (PUT)
    - The BASE disks, holding the component's source and object code
    - The BUILD disks, on which you will build your new system
    - The system's minidisks.

- The third section (introduced by the :PRTFNC. tag) lists the EXECs needed to service each part of the component. There are three functional sections:
  - Receive (introduced by the :RECSER. tag)
  - Apply (introduced by the :APP. tag)
  - Build (introduced by the :BLD. tag).

- Each override area (introduced by an *:overname.* tag) contains changes that can be made to the corresponding component area when the user wishes. The override area permits you to build several different systems with a single product parameter file.

A Sample Product Parameter File

The listing below is a sample product parameter file. The tags are explained in "The Tags in the Product Parameter File" on page 412.

```
*====================================================================
*            Product Parameter File for VM/XA SP Release 2
*====================================================================
*
* The following APAR's have been applied to this file:
*
*  VM32064 - 05/24/88 - New Support for Service Enhancements 2.1
*  VM32436 - 08/01/88 - New Function for Service Enhancements 2.1
*  VM35065 - 01/04/89 - Change to MACUPDT VMFRCUPD in RECSER section
*                       New load lists in CP/CMS in BLD sections
*  VM35315 - 01/06/89 - PE VM35065 Change EXECUPD VMFRCCOM to
*                       EXECUPD VMFRCUPD in RECSER section of DV and GCS
*  VM35316 - 01/06/89 - Remove 19E from SYSTEM disk string
*  VM35405 - 01/09/89 - Remove DELTA2 string from GCS section in $PPF
*  VM36183 - 02/28/89 - Add CPYSYSDV to DV BLD section in $PPF. Requires
*                       Dump View CO-REQ apar VM36326.
*  VM36911 - 04/20/89 - Add HELP to DVF section of PPF. Prereqs DVF
*                       PTF which shipped this service HELP (VM36585)
*  VM37518 - 06/23/89 - New AUX file part handlers added.  New CKUPD tag
*                       added.  Reformatted for readability.  COR
*                       overrides moved to after base sections.
*
*====================================================================
*            NOTE: All tags must be in upper case.
*====================================================================

*====================================================================
* Start of Product Header - List of Components in VM/XA SP Release 2
*====================================================================
:COMPLST.      CP    CMS    DV    GCS
:OVERLSTP.
:OVERLST.   CORCP CORCMS CORDV CORGCS
*====================================================================
* End of Product Header
*====================================================================

*====================================================================
:CP.                        * Start of Parameters for CP
*====================================================================
:CNTRLOP.                   * Section one - Control Options
**                          * DO NOT DELETE THESE CONTROL TAGS
* TAG      VALUE(S)         * DESCRIPTION
**
:CNTRL.    HCPXA            * Control File name
```

```
:UPDTID.   AUXXA              * Update level identifier for building AUX
:SLVI.     H/HP H/PP          * System level and version indicators
:NLS.      AMENG              * System Language
:USEREXIT. VMFUECP            * User exit EXEC.  Called before initial
**                            * accesses and after final access retore.
**                            * 2 paramters passed will indicate calling
**                            * EXEC and SET-UP or CLEAN-UP.
:MERGE.    DELTA1 APPLY LOCAL1
**                            * Disk strings to be merged at Receive
:LTO.      NO                 * Erase lower level TEXT decks at Receive
:CKTXT.    YES                * Check for full text at Apply
:CKUPD.    NO                 * Check update fm regardless of $PTFPART fm
:CKAUX.    YES                * Check composite AUX at Build
:CKREQ.    YES                * Check Requisites at Build
*===================================================================
:MDA.                         * Section two - Mini-Disk Assignments
**                            * STRING NAMES MUST START IN COLUMN 1
* STRNGNAM MINIDISKS           * DESCRIPTION
**
*TASK      191                * Build disks accessed before rest of dbase
LOCAL1     295 591 291        * Disks for corrective and local service
*LOCALn                       * You may define additional LOCAL disks
APPLY      492 192            * Control structure built by service execs
DELTA1     594 294            * CP service from TAPE
*DELTAn                       * You may define additional DELTA disks
BASE1      194                * CP text,loadlist,control file,maclibs
BASE2      394                * CP source,macro defs,and copy files
*BASEn                        * You may define additional BASE disks
*BUILD1    423                * New SYSRES for CP Nucleus Build
BUILD2     49D                * New CMS system disk for AMENG HELP files
BUILD3     49C                * New CMS system disk for UCENG HELP files
BUILD4     490                * New CMS system disk for nucleus etc.
*BUILDn                       * You may define additional BUILD disks
SYSTEM     191 395 691 391 593 293 193
**                            * Disks reqd for your system environment.
**                            * CMS database needed for MACLIBs.  If
**                            * latest MACLIBs are on a build disk, only
**                            * that disk must appear here.
*===================================================================
:PRTFNC.                      * Section three - Part type/Function List
*===================================================================
:RECSER.                      * Receive (Service)
**
* TAPEFILE EXEC     TARGET    * DESCRIPTION
**
  AXLIST   VMFRCAXL DELTA1    * Apply and Exclude lists
  PARTLST  VMFRCCOM DELTA1    * PTF Parts Lists
  UPDT     VMFRCUPD DELTA1    * Update files for CP ASSEMBLE source
  TEXT     VMFRCTXT DELTA1    * Text replacement modules for CP
**                            * Text shells created on target disk
**                            * Update shells created on APPLY disk
  MACAUX   VMFRCAUX DELTA1    * AUX files for updates to non-ASSEM parts
  MACUPDT  VMFRCUPD DELTA1    * Update files for non-ASSEMBLE parts
  SOURCE   VMFRCCOM DELTA1    * New Source (assemble,macro,copy,$exec...)
  MACLIB   VMFRCCOM DELTA1    * Maclibs are not replaced on the base
  EXEC     VMFRCCOM DELTA1    * Replacement parts and regenerated
**                            * parts other than maclibs, text, modules
  HELP     VMFRCCOM BUILD2    * AMENG HELP files go directly to the
**                            * AMENG help Build disk
```

```
  HELP     VMFRCUPP BUILD3   * UCENG HELP files go directly to the
**                           * UCENG help Build disk
  RESV    *VMFRCCOM DELTA1   * Reserved parts
  CPMOD    VMFRCCOM BUILD4   * CP modules targeted for system disk
  PTFS     VMFRCCOM DELTA1   * PTF numbered parts other than TXT
*=====================================================================
:APP.                       * Apply Function
**
* PARTTYPE EXEC     RENAME   * DESCRIPTION
**
  TXT*     VMFAPTXT TEXT     * Create AUX and rename appropriate update
**                           * files to fm5 for names in loadlist.
**                           * Other TXT parts renamed to TEXT
  TAM*     VMFAPNLS TXTAMENG * For American English NLS support
  TKA*     VMFAPNLS TXTKANJI * For KANJI NLS support
  TUC*     VMFAPNLS TXTUCENG * For Uppercase English NLS support
  TPO*     VMFAPNLS TXTPORTG * For Brazilian Portuguese NLS support
  TFR*     VMFAPNLS TXTFRANC * For French NLS support
  TGR*     VMFAPNLS TXTGER   * For German NLS support
  AUXXA    VMFAPAUX          * For MACRO and EXEC AUX files
  AXA*     VMFAPAUX AUXXA    * For MACRO and EXEC AUX files
  EXEC     VMFAPCOM          *
  MACRO    VMFAPCOM          *
** .       VMFAPCOM          * ALL PARTS ARE LISTED BY FILETYPE
** .       VMFAPCOM          * If a part type is not listed
** .       VMFAPCOM          * VMFAPCOM is called by default
*=====================================================================
:BLD.                       * Build Function
**
* BUILDLST EXEC     TARGET   * DESCRIPTION
**
  CPLOAD   VMFBDNUC *BUILD1  * Create CP nucleus based on loadlist
**                           * Target disk symbol agrees with
**                           * disk address in HCPSYS ASSEMBLE
**                           * on the SYSRES macro. Defaults to
**                           * what is in HCPSYS ASSEMBLE if no
**                           * target disk is specified.
**
  CPYSYSCP VMFBDCPY BUILD4   * Copy TEXT decks that do not become
**                           * part of a MODULE to the system disk
**
  CPYBTHCP VMFBDCPY DELTA1   * Copy TEXT decks used by UTILITY and
**                           * listed in the load list, CPLOAD EXEC,
**                           * to the DELTA disk. VMFAPPLY only
**                           * copies the ones not listed in the
**                           * load list.
**
* CPYUTLCP VMFBDCPY DELTA1   * Copy TEXT decks used by UTILITY which
**                           * are not listed in the load list,
**                           * CPLOAD EXEC. This list would be used
**                           * if you have local modifications and
**                           * you choose not to run VMFAPPLY. All
**                           * TEXT decks for which you have received
**                           * service and/or have local mods would
**                           * be copied as a result. You need to remove
**                           * the asterisk before CPYUTLCP before
**                           * running VMFBLD.
**                           * Use UTILITY to build CP Utilities
*=====================================================================
```

```
:END.                          * End of Parameters for CP
*================================================================

*================================================================
:CORCP.  CP                    * Start of CP Corrective Service Overrides
*================================================================
:UPDTID. AUXCOR                * COR service applied at AUXCOR level
:MERGE. LOCAL1                 * Merge only LOCAL1 not APPLY and DELTA1
:MDA. UPDATE
LOCAL1 295 591                 * Merge doesn't go to SUP reach-ahead disk
APPLY  295 591 291 492 192     * LOCAL1 disks added to APPLY string
DELTA1 295 591 291 594 294     * LOCAL1 disks added to DELTA1 string
*================================================================
:END.                          * End of CP Corrective Service Overrides
*================================================================

*================================================================
:CMS.                          * Start of Parameters for CMS
*================================================================
:CNTRLOP.                      * Section one - Control Options
**                             * DO NOT DELETE THESE CONTROL TAGS
* TAG       VALUE(S)          * DESCRIPTION
**
:CNTRL.     DMSXA              * Control File name
:UPDTID.    AUXXA              * Update level identifier for building AUX
:SLVI.      H/DS T/DS          * System level and version indicators
:NLS.       AMENG              * System Language
:USEREXIT.  VMFUECMS           * User exit EXEC.  Called before initial
**                             * accesses and after final access retore.
**                             * 2 paramters passed will indicate calling
**                             * EXEC and SET-UP or CLEAN-UP.
:MERGE.     DELTA1 APPLY LOCAL1
**                             * Disk strings to be merged at Receive
:LTO.       NO                 * Erase lower level TEXT decks at Receive
:CKTXT.     YES                * Check for full text at Apply
:CKUPD.     NO                 * Check update fm regardless of $PTFPART fm
:CKAUX.     YES                * Check composite AUX at Build
:CKREQ.     YES                * Check Requisites at Build
*================================================================
:MDA.                          * Section two - Mini-Disk Assignments
**                             * STRING NAMES MUST START IN COLUMN 1
* STRNGNAM MINIDISKS           * DESCRIPTION
**
*TASK       191                * Build disks accessed before rest of dbase
LOCAL1      395 691 391        * Disks for corrective and local service
*LOCALn                        * You may define additional LOCAL disks
APPLY       692 392            * Control structure built by service execs
DELTA1      593 293            * CMS service from TAPE
*DELTAn                        * You may define additional DELTA disks
BASE1       193                * CMS text,loadlist,control file,maclibs
BASE2       393                * CMS source and macro definitions
*BASEn                         * You may define additional BASE disks
BUILD1      490                * New CMS system disk for nucleus etc.
BUILD2      49D                * New CMS system disk for AMENG HELP files
BUILD3      501                * New CMS system disk for EREP files
BUILD4      49C                * New CMS system disk for UCENG HELP files
*BUILDn                        * You may define additional BUILD disks
SYSTEM      191 295 591 291 594 294 194
**                             * Disks reqd for your system environment.
**                             * CP database needed for MACLIBs.  If
```

```
**                          * latest MACLIBs are on a build disk, only
**                          * that disk must appear here.
*=====================================================================
:PRTFNC.                    * Section three - Part type/Function List
*=====================================================================
:RECSER.                    * Receive (Service)
**
* TAPEFILE EXEC     TARGET   * DESCRIPTION
**
  AXLIST   VMFRCAXL DELTA1   * Apply and Exclude lists
  PARTLST  VMFRCCOM DELTA1   * PTF Parts Lists
  UPDT     VMFRCUPD DELTA1   * Update files for CMS ASSEMBLE source
  TEXT     VMFRCTXT DELTA1   * Text replacement modules for CMS
**                          * Text shells created on target disk
**                          * Update shells created on APPLY disk
  MACAUX   VMFRCAUX DELTA1   * AUX files for updates to non-ASSEM parts
  MACUPDT  VMFRCUPD DELTA1   * Update files for non-ASSEMBLE parts
  SOURCE   VMFRCCOM DELTA1   * New Source (assemble,macro,copy,$exec...)
  MACLIB   VMFRCCOM BUILD1   * Maclibs go directly to the Build disk
  EXEC     VMFRCCOM DELTA1   * Replacement parts and regenerated
**                          * parts other than maclibs, text, modules
  MODULE   VMFRCCOM BUILD1   * CMS/IOCP modules go directly to the
**                          * Build disk
  HELP     VMFRCCOM BUILD2   * AMENG HELP files go directly to the
**                          * AMENG help Build disk
  HELP     VMFRCUPP BUILD4   * UCENG HELP files go directly to the
**                          * UCENG help Build disk
  IOCP     VMFRCTXT DELTA1   * IOCP files go to the DELTA1 disk
  RESV    *VMFRCCOM DELTA1   * Reserved parts
  PTFS     VMFRCCOM DELTA1   * ptf numbered parts other than TXT
*=====================================================================
:APP.                       * Apply Function
**
* PARTTYPE EXEC     RENAME   * DESCRIPTION
**
  TXT*     VMFAPTXT TEXT     * Create AUX and rename appropriate update
**                          * files to fm5 for names in loadlist.
**                          * Other TXT parts renamed to TEXT
  TAM*     VMFAPNLS TXTAMENG * For American English NLS support
  TKA*     VMFAPNLS TXTKANJI * For KANJI NLS support
  TUC*     VMFAPNLS TXTUCENG * For Uppercase English NLS support
  TPO*     VMFAPNLS TXTPORTG * For Brazilian Portuguese NLS support
  TFR*     VMFAPNLS TXTFRANC * For French NLS support
  TGR*     VMFAPNLS TXTGER   * For German NLS support
  AUXXA    VMFAPAUX          * For EXEC AUX files
  AXA*     VMFAPAUX AUXXA    * For EXEC AUX files
  AUXMXA   VMFAPAUX          * For MACRO AUX files
  AXM*     VMFAPAUX AUXMXA   * For MACRO AUX files
  EXEC     VMFAPCOM          *
  MACRO    VMFAPCOM          *
** .       VMFAPCOM          * ALL PARTS ARE LISTED BY FILETYPE
** .       VMFAPCOM          * If a part type is not listed
** .       VMFAPCOM          * VMFAPCOM is called by default
*=====================================================================
:BLD.                       * Build Function
**
* BUILDLST EXEC     TARGET   * DESCRIPTION
**
  CMSLOAD  VMFBDNUC BUILD1   * Create CMS nucleus based on loadlist
**
```

```
  CPYSYSCM VMFBDCPY BUILD1   * Copy TEXT decks that do not become
**                           * part of a MODULE to the system disk
**
  CPYBTHCM VMFBDCPY DELTA1   * Copy TEXT decks used by CMSGEND and
**                           * listed in the load list, CMSLOAD EXEC,
**                           * to the DELTA disk. VMFAPPLY only
**                           * copies the ones not listed in the load
**                           * list.
**
* CPYGNDCM VMFBDCPY DELTA1   * Copy TEXT decks used by CMSGEND which
**                           * are not listed in the load list,
**                           * CMSLOAD EXEC. This list would be used
**                           * if you have local modifications and
**                           * you choose not to run VMFAPPLY. All
**                           * TEXT decks for which you have received
**                           * service and/or have local mods would
**                           * be copied as a result. You need to remove
**                           * the asterisk before CPYGNDCM before
**                           * running VMFBLD.
**                           * Use CMSGEND to build CMS modules
*=====================================================================
:END.                        * End of Parameters for CMS
*=====================================================================

*=====================================================================
:CORCMS. CMS                 * Start of CMS Corrective Service Overrides
*=====================================================================
:UPDTID. AUXCOR              * COR service applied at AUXCOR level
:MERGE. LOCAL1               * Merge only LOCAL1 not APPLY and DELTA1
:MDA. UPDATE
LOCAL1 395 691               * Merge doesn't go to SUP reach-ahead disk
APPLY  395 691 391 692 392   * LOCAL1 disks added to APPLY string
DELTA1 395 691 391 593 293   * LOCAL1 disks added to DELTA1 string
*=====================================================================
:END.                        * End of CMS Corrective Service Overrides
*=====================================================================

*=====================================================================
:DV.                         * Start of Parameters for DV
*=====================================================================
:CNTRLOP.                    * Section one - Control Options
**                           * DO NOT DELETE THESE CONTROL TAGS
* TAG       VALUE(S)         * DESCRIPTION
**
:CNTRL.     HCSXA            * Control File name
:UPDTID.    AUXXA            * Update level identifier for building AUX
:SLVI.      H/HP H/PP        * System level and version indicators
:NLS.       AMENG            * System Language
:USEREXIT.  VMFUEDV          * User exit EXEC.  Called before initial
**                           * accesses and after final access retore.
**                           * 2 paramters passed will indicate calling
**                           * EXEC and SET-UP or CLEAN-UP.
:MERGE.     DELTA1 APPLY LOCAL1
**                           * Disk strings to be merged at Receive
:LTO.       NO               * Erase lower level TEXT decks at Receive
:CKTXT.     YES              * Check for full text at Apply
:CKUPD.     NO               * Check update fm regardless of $PTFPART fm
:CKAUX.     YES              * Check composite AUX at Build
:CKREQ.     YES              * Check Requisites at Build
*=====================================================================
```

```
:MDA.                          * Section two - Mini-Disk Assignments
**                             * STRING NAMES MUST START IN COLUMN 1
* STRNGNAM MINIDISKS           * DESCRIPTION
**
*TASK      191                 * Build disks accessed before rest of dbase
LOCAL1     395 691 391         * Disks for corrective and local service
*LOCALn                        * You may define additional LOCAL disks
APPLY      692 392             * Control structure built by service execs
DELTA1     593 293             * DV service from TAPE
*DELTAn                        * You may define additional DELTA disks
BASE1      193                 * DV text,loadlist,control file,maclibs
*BASEn                         * You may define additional BASE disks
BUILD1     490                 * New system disk for CMS and Dump Viewer
BUILD2     49D                 * New CMS system disk for AMENG HELP files
BUILD4     49C                 * New CMS system disk for UCENG HELP files
*BUILDn                        * You may define additional BUILD disks
SYSTEM     191 295 591 291 594 294 194
**                             * Disks reqd for your system environment.
**                             * CP database needed for MACLIBs.  If
**                             * latest MACLIBs are on a build disk, only
**                             * that disk must appear here.
*================================================================
:PRTFNC.                       * Section three - Part type/Function List
*================================================================
:RECSER.                       * Receive (Service)
**
* TAPEFILE EXEC     TARGET    * DESCRIPTION
**
  AXLIST   VMFRCAXL DELTA1     * Apply and Exclude lists
  PARTLST  VMFRCCOM DELTA1     * PTF Parts Lists
  UPDT     VMFRCUPD DELTA1     * Update files for HCSTBL ASSEMBLE source
  DVFTEXT  VMFRCTXT DELTA1     * Text replacement modules for DV
**                             * Text shells created on target disk
**                             * Update shells created on APPLY disk
  EXECAUX  VMFRCAUX DELTA1     * AUX files for updates to non-ASSEM parts
  EXECUPD  VMFRCUPD DELTA1     * XEDIT updates to DELTA disk
**                             * (DVFXEDIT $XEDIT and FINDUSER $XEDIT)
  SOURCE   VMFRCCOM DELTA1     * New Source (assemble,macro,copy,$exec...)
  DVFEXEC  VMFRCCOM BUILD1     * DUMP VIEWING EXECs go directly
**                             * to the Build disk
**                             * includes CPBLOCK CBMAP and HCSCP1 TABLE
  DVFMODS  VMFRCCOM BUILD1     * DUMP VIEWING modules go directly
**                             * to the Build disk
  HELP     VMFRCCOM BUILD2     * AMENG HELP files go directly to the
**                             * AMENG help Build disk
  HELP     VMFRCUPP BUILD4     * UCENG HELP files go directly to the
**                             * UCENG help Build disk
  RESV    *VMFRCCOM DELTA1     * Reserved parts
  PTFS     VMFRCCOM DELTA1     * PTF numbered parts other than TXT
*================================================================
:APP.                          * Apply Function
**
* PARTTYPE EXEC     RENAME    * DESCRIPTION
**
  TXT*     VMFAPTXT TEXT       * Create AUX and rename appropriate update
**                             * files to fm5 for names in loadlist.
**                             * Other TXT parts renamed to TEXT
  TAM*     VMFAPNLS TXTAMENG * For American English NLS support
  TKA*     VMFAPNLS TXTKANJI * For KANJI NLS support
  TUC*     VMFAPNLS TXTUCENG * For Uppercase English NLS support
```

```
  TPO*     VMFAPNLS TXTPORTG * For Brazilian Portuguese NLS support
  TFR*     VMFAPNLS TXTFRANC * For French NLS support
  TGR*     VMFAPNLS TXTGER   * For German NLS support
  AUXXA    VMFAPAUX          * For EXEC AUX files
  AXA*     VMFAPAUX AUXXA    * For EXEC AUX files
  EXEC     VMFAPCOM          *
** .       VMFAPCOM          * ALL PARTS ARE LISTED BY FILETYPE
** .       VMFAPCOM          * If a part type is not listed
** .       VMFAPCOM          * VMFAPCOM is called by default
*=====================================================================
:BLD.                       * Build Function
**
* BUILDLST EXEC     TARGET   * DESCRIPTION
**
  CPYSYSDV VMFBDCPY BUILD1   * Copy DV Interface Files to System disk
**
**                          * Use UTILITY DVFGEND to build Dump Viewer
*=====================================================================
:END.                       * End of Parameters for DV
*=====================================================================

*=====================================================================
:CORDV.  DV                 * Start of DV Corrective Service Overrides
*=====================================================================
:UPDTID. AUXCOR             * COR service applied at AUXCOR level
:MERGE. LOCAL1              * Merge only LOCAL1 not APPLY and DELTA1
:MDA. UPDATE
LOCAL1 395 691              * Merge doesn't go to SUP reach-ahead disk
APPLY  395 691 391 692 392  * LOCAL1 disks added to APPLY string
DELTA1 395 691 391 593 293  * LOCAL1 disks added to DELTA1 string
*=====================================================================
:END.                       * End of DV Corrective Service Overrides
*=====================================================================

*=====================================================================
:GCS.                       * Start of Parameters for GCS
*=====================================================================
:CNTRLOP.                   * Section one - Control Options
**                          * DO NOT DELETE THESE CONTROL TAGS
* TAG      VALUE(S)         * DESCRIPTION
**
:CNTRL.    CSIXA            * Control File name
:UPDTID.   AUXXA            * Update level identifier for building AUX
:SLVI.     H/CI             * System level and version indicators
:NLS.      AMENG            * System Language
:USEREXIT. VMFUECMS         * User exit EXEC.  Called before initial
**                          * accesses and after final access retore.
**                          * 2 paramters passed will indicate calling
**                          * EXEC and SET-UP or CLEAN-UP.
:MERGE.    DELTA1 APPLY LOCAL1
**                          * Disk strings to be merged at Receive
:LTO.      NO               * Erase lower level TEXT decks at Receive
:CKTXT.    YES              * Check for full text at Apply
:CKUPD.    NO               * Check update fm regardless of $PTFPART fm
:CKAUX.    YES              * Check composite AUX at Build
:CKREQ.    YES              * Check Requisites at Build
*=====================================================================
:MDA.                       * Section two - Mini-Disk Assignments
**                          * STRING NAMES MUST START IN COLUMN 1
* STRNGNAM MINIDISKS        * DESCRIPTION
```

```
**
*TASK      191               * Build disks accessed before rest of dbase
LOCAL1     495 791 491       * Disks for corrective and local service
*LOCALn                      * You may define additional LOCAL disks
APPLY      892 592           * Control structure built by service execs
DELTA1     896 596           * GCS service from TAPE
*DELTAn                      * You may define additional DELTA disks
BASE1      595               * GCS text,loadlist,control file,maclibs
**                           * Nucleus, LOADLIB, EXECs
*BASE2                       * GCS source files ASSEMBLE,$EXEC,$XEDIT
*BASEn                       * You may define additional BASE disks
BUILD1     895               * New GCS system disk for nucleus etc.
BUILD2     89E               * New GCS system disk extension
BUILD3     49D               * New CMS system disk for AMENG HELP files
BUILD4     49C               * New CMS system disk for UCENG HELP files
BUILD5     490               * New CMS system disk for GCS Interface
*BUILDn                      * You may define additional BUILD disks
SYSTEM     191 395 691 391 593 293 193
**                           * Disks reqd for your system environment.
**                           * CMS database needed for MACLIBs.  If
**                           * latest MACLIBs are on a build disk, only
**                           * that disk must appear here.
*============================================================================
:PRTFNC.                     * Section three - Part type/Function List
*============================================================================
:RECSER.                     * Receive (Service)
**
* TAPEFILE EXEC     TARGET   * DESCRIPTION
**
  AXLIST   VMFRCAXL DELTA1    * Apply and Exclude lists
  PARTLST  VMFRCCOM DELTA1    * PTF Parts Lists
  TEXT     VMFRCTXT DELTA1    * Text replacement modules for GCS
**                           * Text shells created on target disk
**                           * Update shells created on APPLY disk
  EXECAUX  VMFRCAUX DELTA1    * AUX files for updates to non-ASSEM parts
  EXECUPD  VMFRCUPD DELTA1    * Update files for non-ASSEMBLE parts
  SOURCE   VMFRCCOM DELTA1    * NEW SOURCE FILES GO ON DELTA1
  MACLIB   VMFRCCOM BUILD1    * Maclibs go directly to the Build disk
  EXEC     VMFRCCOM DELTA1    * Replacement parts and regenerated
**                           * parts other than maclibs, text, modules
  MODULE   VMFRCCOM BUILD1    * GCS modules go directly to the Build disk
  HELP     VMFRCCOM BUILD3    * AMENG HELP files go directly to the
**                           * AMENG help Build disk
  HELP     VMFRCUPP BUILD4    * UCENG HELP files go directly to the
**                           * UCENG help Build disk
  RESV    *VMFRCCOM DELTA1    * Reserved parts
  PTFS     VMFRCCOM DELTA1    * PTF numbered parts other than TXT
  GCSCMS   VMFRCTXT DELTA1    * Text replacement GCS interface files.
**                           * Text shells created on target disk
**                           * Update shells created on APPLY disk
*============================================================================
:APP.                        * Apply Function
**
* PARTTYPE EXEC     RENAME   * DESCRIPTION
**
  TXT*     VMFAPTXT TEXT      * Create AUX and rename appropriate update
**                           * files to fm5 for names in loadlist.
**                           * Other TXT parts renamed to TEXT
  TAM*     VMFAPNLS TXTAMENG * For American English NLS support
  TKA*     VMFAPNLS TXTKANJI * For KANJI NLS support
```

```
  TUC*     VMFAPNLS TXTUCENG * For Uppercase English NLS support
  TPO*     VMFAPNLS TXTPORTG * For Brazilian Portuguese NLS support
  TFR*     VMFAPNLS TXTFRANC * For French NLS support
  TGR*     VMFAPNLS TXTGER   * For German NLS support
  AUXXA    VMFAPAUX          * For MACRO and EXEC AUX files
  AXA*     VMFAPAUX AUXXA    * For MACRO and EXEC AUX files
  EXEC     VMFAPCOM          *
  MACRO    VMFAPCOM          *
** .       VMFAPCOM          * ALL PARTS ARE LISTED BY FILE TYPE
** .       VMFAPCOM          * If a part type is not listed
** .       VMFAPCOM          * VMFAPCOM is called by default
*=====================================================================
:BLD.                       * Build Function
**
* BUILDLST EXEC     TARGET   * DESCRIPTION
**
  GCSLOAD  VMFBDNUC BUILD1   * Create GCS NUCLEUS based on LOADLIST
**                           * Use GROUP EXEC to build GCS Macilbs
**
  CPYGCS   VMFBDCPY BUILD5   * Rename/Copy GCS Interface Files, needed
**                           * for DUMPSCAN formatters, these are
**                           * TEXT decks.
**
* CPYOTHGC VMFBDCPY DELTA1   * PMX modules or TEXT decks, message
**                           * exchangers, loaded in SNA program
**                           * products, that do not become part of
**                           * the GCS nucleus. If you choose not
**                           * to run VMFAPPLY, this list would
**                           * copy these text decks to the
**                           * DELTA disk during the VMFBLD process.
*=====================================================================
:END.                       * End of Parameters for GCS
*=====================================================================

*=====================================================================
:CORGCS. GCS                * Start of GCS Corrective Service Overrides
*=====================================================================
:UPDTID. AUXCOR             * COR service applied at AUXCOR level
:MERGE. LOCAL1              * Merge only LOCAL1 not APPLY and DELTA1
:MDA. UPDATE
LOCAL1 495 791              * Merge doesn't go to SUP reach-ahead disk
APPLY  495 791 491 892 592  * LOCAL1 disks added to APPLY string
DELTA1 495 791 491 896 596  * LOCAL1 disks added to DELTA1 string
*=====================================================================
:END.                       * End of GCS Corrective Service Overrides
*=====================================================================
```

Warning: The disk addresses in the SPLOAD PROFILE, the user directory, 56643082 $PPF, and the SETUP EXEC must all match the default addresses the ITASK EXEC uses. If you change an address in any one of these places, you must change it in all the others. We recommend that you do not change any addresses.

The Tags in the Product Parameter File

Product Area

:COMPLST. *compname1 compname2 ... compnamen comment*

:COMPLST.
is a list of the components in the product. It can be specified only in the product area of the base product parameter file (where it is required), not in an override file.

compname
is an unofficial nickname, in upper case, for the component. The first component of the product on the PUT gets the first compname in the list, the second component the second compname and so on.

comment
is any string prefixed with an asterisk. A comment can appear after any of the tags or on a line by itself anywhere in the product parameter file.

:OVERLSTP. *overname1 overname2 ... overnamen*
is a list of override names that you want VMFOVER to prompt you with, in addition to the list of component names following the :COMPLST. tag, if VMFOVER is invoked without a component being specified. If you do not want VMFOVER to prompt you with any override names, specify :OVERLSTP. followed by an empty list.

If you do not specify the :OVERLSTP. tag, the names following the :OVERLST. tag are used for prompting. You do not have to list the override names twice. You will be prompted only once for any name listed on both tags.

The :OVERLSTP. tag is required in the product area of the base product parameter file and permitted in the product area of a separate override file. It is not allowed in the component override section of either file.

:OVERLSTP. must always be placed between the :COMPLST. tag and the :OVERLST. tag in the PPF.

:OVERLST. *overname1 overname2 ... overnamen*
is a list of override tags, each one related to a specific component. Several override tags can relate to the same component, but each override tag can only relate to one component.

If you do not specify the :OVERLSTP. tag, the names following the :OVERLST. tag are used for prompting. You do not have to list the override names twice. You will be prompted only once for any name listed on both tags.

The :OVERLST. specification is required in the product area of both the base product parameter file and the separate override file. It is not allowed in the component override section of either file.

:OVERLST. must always be placed after the :OVERLSTP. tag in the base PPF.

Component Area

:*compname.*
is an uppercase delimiter for the part of the PPF describing the component named. *compname* is one of the components named in :COMPLST. The end of the component part of the PPF for this component is delimited by a corresponding :END. tag.

Control Options

:CNTRLOP.
is the delimiter for the component section of the PPF defining general parameters for the component.

:CNTRL. *control_file_name*
is the filename of the control file to be used to service, assemble, and/or build this component of the product.

:SLVI. *p1/ss1 p2/ss2 ... pn/ssn*

:SLVI
is a list of the system level and version indicators used as a prefix and suffix, concatenated with the numeric portion of the APAR number, to form the filetype of the update file.

p/ss
is a word that consists of two values separated by a slash. The first value, represented by a p, is the single character prefix in the update file filetype. The second value, represented by ss, is the 2-character suffix in the update file filetype.

:NLS. *langid*
is specified to be one of the following: AMENG, KANJI, UCENG, PORTG, FRANC, or GER. For CP and CMS the default is AMENG.

:LTO. YES|NO

:LTO.
The Last Text Only option erases lower level text decks during receive. Duplicate text decks are never created, regardless of the setting of the LTO option (yes or no).

YES
indicates that during VMFREC a search is made for other text decks with the same filename and filetype prefix in the target minidisk string. If other decks exist, the PTF information in each text deck prologue is checked and only the latest text deck, including the one just read in, is kept on the DELTA disk. The latest text deck includes the accumulation of all service in the decks being replaced. If this is not the case an error is included in the receive exception log. This requires write access to all DELTA disks on which text decks may exist.

NO
indicates that existing text decks for different PTFs are left as they are. If a text deck exists for the same PTF the new one is not kept.

:CKAUX. YES|NO

:CKAUX.
Check composite AUX at build. The value of this tag may be overriden by using the CHECK or NOCHECK option on VMFBLD. CHECK is equivalent to :CKAUX. YES and NOCHECK is equivalent to :CKAUX. NO.

YES
indicates that all AUX and update entries in the text deck should be verified against the AUX control structure to be sure the correct deck has been located.

NO
indicates that only the entries in the AUX file used to locate the text deck will be verified against the text deck prolog. Additional entries in the text deck prolog are not verified for faster completion of the build process.

:CKREQ. YES|NO

:CKREQ.
Check requisites at build. The value of this tag may be overridden by using the CHECK or NOCHECK option on VMFBLD. CHECK is equivalent to :CKREQ. YES and NOCHECK is equivalent to :CKREQ. NO.

YES
indicates that requisite information chains will be verified to assure that all required PTFs related to the ones applied are included in the system being built. The update files contain entries for PREREQ, CO-REQ and IF-REQ.

PREREQs and CO-REQs describe dependencies in the same product. If the dependency is in the same component, the 7-character APAR number is listed (for example, VM12345). If the dependency is in another component of the same product the APAR number and component ID are included in parentheses following the PTF number, for example, UV54321(VM12345-56630130).

If the dependency is in another product, an IF-REQ is used. An IF-REQ contains a PTF number rather than an APAR number. The APAR number and component ID are included in parentheses following the PTF number, for example, UV54321(VM12345-56630130).

NO
indicates that requisite information chains as contained in the text will not be verified during the VMFBLD EXEC.

:CKUPD. YES|NO

:CKUPD.
Check and apply PTFs whether or not they have already been applied. A filemode of 5 on the $PTFPART list indicates that a PTF has been applied.

YES
indicates that a filemode of 5 on a $PTFPART list is not sufficient to prevent a PTF from being applied. Additional processing is performed for each part in the list. This makes it possible for the same PTF to be processed more than once.

NO
indicates that filemode 5 on the $PTFPART list will prevent the same PTF from being processed more than once. If the PTF has already been successfully applied as indicated by its filemode all further processing for this PTF is bypassed.

:CKTXT. YES|NO

:CKTXT.
Check for full text at apply. The value of this tag may be overridden by using the CHECK or NOCHECK option on VMFAPPLY. CHECK is equivalent to :CKTXT. YES. NOCHECK is equivalent to :CKTXT. NO.

YES
indicates that when VMFAPPLY is complete, if a PTF which has been applied is a text shell and not a full text deck including executable code, an error should be placed in the Apply Exception Log.

NO
indicates that the application of a PTF which is a text shell will not cause an error in the exception log.

:UPDTID. *updateid*
identifies the filetype of an AUX file and should match a filetype listed in the control file. This is the level at which AUX files will be updated or created as PTFs are applied. This updateid will be used unless you override it by providing an updateid as an argument to VMFAPPLY.

:USEREXIT. *exitname*

:USEREXIT.
identifies the user EXEC for setup and cleanup to be called at beginning and end of each service function (VMFREC, VMFAPPLY, and so on).

exitname
is the name of the user EXEC to be called. The first parameter is the uppercase name of the invoking EXEC. The second parameter is either SET-UP or CLEAN-UP, indicating initialization or termination. At termination the third parameter is the return code as determined by the invoking EXEC. The return code can be changed by the exit. Functions such as link and detach can be invoked by the exit. VMFUECP, VMFUECMS, and VMFUEDV are the default exits provided by IBM. These exits are no-ops.

:MERGE. *symbol1 symbol2 ... symboln*

:MERGE.
is a list of symbolic disk strings (for example DELTA, APPLY) to be merged by VMFREC when the product or component being received already exists on the receive target disk. If the target disk does not contain the product or component being received, the merge does not take place.

VMFREC accomplishes the merge by moving the contents of the disk before the last in the string to the last disk in the string, then erasing the disk before the last. It repeats this process with the next pair of disks, moving one disk to the left in the string each time, until the first disk in the string is clear.

Note: If you want to prevent merging from taking place, see Appendix G, "Controlling Disk String Merges" on page 859.

symbol
is the symbolic name of a minidisk string such as DELTA1, APPLY, or LOCAL1. If the product or component is present, as indicated in a file with the filetype $MR*xxxxx*, on the first disk in any disk string in the list, all disk strings in the list are merged.

Minidisk Assignments:

Usage Notes:

- Considerations for read-only disks in the product parameter file:

  Disks appearing in the MDA section of the $PPF may be specified as read only by coding a slash (/) immediately following the disk address in the symbolic disk string. For example,

  ```
  LOCAL 295 591/ 291
  ```

  indicates that the 591 disk is to be read only and it will be accessed as mode/mode by VMFSETUP.

- The last specification wins:

  If the disk address appears in a given section (receive, apply, etc.) more than once, its last specification determines its access status.

- Disks currently accessed R/W:

  If a disk is currently accessed R/W it is released and then accessed mode/mode. Upon performing a VMFSETUP restore it will once again be R/W.

  If the disk is R/W and empty (no CMS files on it) the mode/mode fails and a message is issued when the using application program attempts to access it.

- Merge considerations:

  Do not use read only disks in symbolic disk strings identified as strings to be merged. Merging requires write access.

:MDA.
is an uppercase delimiter for the section describing the minidisk assignments.

TASK *dddd1 dddd2 ... ddddn*

TASK
lists the service EXECs build disk. The TASK disk is accessed after the A-disk and before the component data base disks. For more information on using the TASK statement to set up a separate disk for the service EXECs, see the "Usage Notes" section of the VMFREC EXEC description (on page 710). The symbolic disk name TASK must start in column 1.

dddd
is the 4-digit disk address of a system minidisk.

SYSTEM *dddd1 dddd2 ... ddddn*

SYSTEM
lists the disks required for your working system environment. The symbolic disk name SYSTEM must start in column 1.

dddd
is the 4-digit disk address of a system minidisk.

BASE1 dddd
⋮
BASE*n dddd*

BASE1 ... BASE*n*
are the BASE disks created at installation time. You may define additional BASE disks if you have specific needs. The symbolic disk name BASE must start in column 1.

dddd
is the 4-digit disk address of a BASE minidisk.

DELTA1 *dddd1 dddd2 ... ddddn*
⋮
DELTA*n dddd1 dddd2 ... ddddn*

DELTA1 ... DELTA*n*
are disks containing service from the PUT. You may define additional DELTA disks. The symbolic disk name DELTA must start in column 1.

dddd
is the 4-digit disk address of a DELTA minidisk.

APPLY *dddd1 dddd2 ... ddddn*

APPLY
identifies the control structure to be built. AUX files and update shells are created on the APPLY minidisks. There is only one symbolic string of APPLY minidisks. The symbolic disk name APPLY must start in column 1.

dddd
is the 4-digit disk address of an APPLY minidisk.

LOCAL1 *dddd1 dddd2 ... ddddn*
⋮
LOCAL*n dddd1 dddd2 ... ddddn*

LOCAL1 ... LOCAL*n*
are disks for local service and corrective service. You may define additional LOCAL disks The symbolic disk name LOCAL must start in column 1.

dddd
is the 4-digit disk address of a LOCAL minidisk.

BUILD1 *dddd*
⋮
BUILD*n dddd*

BUILD1 ... BUILD*n*
are the new system residence disks for CP nucleus build. You may define additional BUILD disks. The symbolic disk name BUILD must start in column 1.

Warning: The :BUILD1. tag does not force the nucleus to be built on the specified disk. Make sure that the HCPSYS ASSEMBLE file (for CP), the DMSNGP profile (for CMS), or the saved system information panel (GRP121, for GCS), indicates the disk where you want to build the nucleus.

dddd
is the 4-digit disk address of a BUILD minidisk.

EXECs

:PRTFNC.
is an uppercase delimiter for the section describing the list of the types of parts serviced with the functional EXEC assignments and symbolic target disk assignments.

:RECSER.
is an uppercase delimiter for the list of functional assignments related to the Receive function (VMFREC EXEC). This tag is followed by a list which contains one entry for each possible tape file for the component on the PUT or corrective service tape.

Each tape file contains all the parts of the same or a related type. The attribute that binds the types of part in a particular tape file is that they all require the same Receive processing and target minidisk symbol. Each part is in a separate CMS file within the tape file.

tapfil execname target

tapfil
is an uppercase mnemonic that symbolizes the content of the tape file it represents. This is the first symbol in each entry for each tape file on the PUT or corrective service tape. This symbol may start in any column.

execname
is the name of the EXEC that processes the types of parts found in the tape file represented by this entry. IBM provides four part handlers for the Receive function. VMFRCUPD processes update files. VMFRCTXT processes text decks. VMFRCAXL processes apply and exclude lists. VMFRCCOM processes all other parts by simply putting them to the target disk.

An EXEC name starting with an asterisk causes the associated tape file not to be received.

target
is an uppercase symbol corresponding to the minidisk assignment section of the PPF that identifies where the parts are to be placed as they are received from the PUT.

:APP.
is an uppercase delimiter for the list of functional assignments related to the apply function (VMFAPPLY EXEC). This tag is followed by a list which contains one entry for each type of part for the component.

partype execname rename

partype
is an uppercase part filetype or part filetype prefix when terminated by an asterisk. If the partype symbol is TXT*, all parts with a filetype starting with the characters TXT are processed by the part handler name in this entry. If the symbol is MACRO, all parts with the filetype MACRO are processed by the part handler name in this entry. This symbol may start in any column.

execname
is the name of the EXEC that processes the types of parts found with the filetype represented in this entry. Parts with a filetype for which no partype symbol appears in the list will be processed by the common part type handler VMFAPCOM. Two other part handlers are provided by IBM for the Apply function. VMFAPTXT processes text decks, VMFAPNLS language versions of text decks.

rename
is the new filetype to be given to those parts found to have an existing filetype matching the partype symbol in the entry, but not supported by VMFBLD. Parts supported by VMFBLD are not renamed. Whether parts are supported is determined by finding the part in the build list. Parts in the build list are located by PTF number. Parts not found in the build list are copied with a new filetype of *rename*.

:BLD.
is an uppercase delimiter for the list of functional assignments related to the build function (VMFBLD EXEC). This tag is followed by a list which contains one entry for each type of part which is built from a collection of other parts in the component. This symbol may start in any column.

bldlst execname target

bldlst
is an uppercase filename for the list (build list) of parts to be included in the part being built. For a nucleus this is a load list.

execname
is the filename of the EXEC that processes the build list.

target
is the symbolic name corresponding to the minidisk assignment section of the PPF that identifies where the part being built is to be placed. The minidisk address assigned to the symbol specified here should match the SYSRES address specified on the SYSRES macro in the HCPSYS file.

:END.
identifies the end of the current component part. The next component in the PPF starts with the next *:compname.* from the :COMPLST. tag.

Override Area

:overname. compname ppfname **[NEWNAME | OLDNAME]**

:overname.
identifies the beginning of the override section for a component. This statement does not appear in the temporary PPF.

compname
is the name of the component specified in the base PPF :COMPLST. that is to be overridden.

ppfname
is the name of the PPF that is to be overridden. If *ppfname* is not specified, *overname* and *compname* must be in the same file (that is, the component override is part of the base PPF).

NEWNAME
indicates that the filename of the temporary PPF should be the filename of the override file. This option cannot be used in the override section of the base PPF (that is, when override and base PPF are in the same file).

OLDNAME
indicates that the filename of the temporary PPF should be the original filename of the base PPF file. OLDNAME is the default if neither OLDNAME or NEWNAME is specified.

:MDA. [REPLACE | UPDATE]
:RECSER. [REPLACE | UPDATE]
:APP. [REPLACE | UPDATE]
:BLD. [REPLACE | UPDATE]

:MDA.
introduces the overrides for the minidisk assignment section of the component area.

:RECSER.
introduces the overrides for the list of functional assignments related to the receive function (VMFREC EXEC).

:APP.
introduces the overrides for the list of functional assignments related to the Apply function (VMFAPPLY EXEC).

:BLD.
introduces the overrides for the list of functional assignments related to the build function (VMFBLD EXEC).

UPDATE
indicates that the entries following the tag in the override section will overlay the corresponding entry in the base PPF. Any overlaid entry appears as a comment in the temporary PPF. If the entry does not exist in the base, the entry will be added at the end of the section in the base PPF. UPDATE is the default if neither UPDATE or REPLACE is specified.

REPLACE
indicates that all entries in the base PPF are to be removed and replaced by the entries in the override PPF. The removed entries do not appear as comments in the temporary PPF.

*

Comments or a group of comments beginning with a single asterisk are merged into the temporary PPF immediately preceding the next non-comment statement.

**

Comments or a group of comments beginning with a double asterisk are merged into the temporary PPF immediately following the previous non-comment statement.

***

Comments beginning with three asterisks do not appear in the temporary override file.

**** ...

Comments beginning with more than three asterisks are handled as if they began with one asterisk.

The Product Parameter Override File

The product parameter override file is a separate file containing tags that you want to override the tags in the base product parameter file when you build or service your system. The override file permits you to create a new version of the product without creating a complete new product parameter file.

The only difference between a separate override file and the override section of a base PPF is that the separate override file begins with an optional :OVERLSTP. tag and an :OVERLST. tag. In a base PPF, these tags come immediately after the :COMPLST. tag at the beginning of the file, not in the override section.

The filename of the override file is the name of the new version of the product. The filetype is $PPF.

The following example shows the override file used with the base file 56643082 $PPF to create an alternate product identifier (or "alias") for VM/XA SP. Since the alternate product identifier is VMXA, the override file is called VMXA $PPF.

Note: If you want to invoke any overrides from the override section of the base product parameter file, for example, the corrective service overrides, you must copy them to the separate override file.

Save override files on the CMS alternate LOCAL1 (395) disk. IBM suggests that you should also save the updated product parameter file there.

Figure 29 is an example of a product parameter override file.

```
* This is the alias override for the VM/XA Product 56643082
:OVERLST. CP CMS DV GCS
*** The overlst statement will not appear on the temporary composite
*** product parameter file.
:CP.       CP 56643082       * Get the base unmodified PPF from the CP
***                          * file named 56643082 $PPF
:END.                        * End of current component section
:CMS. CMS 56643082 NEWNAME   * Get the base unmodified PPF from the CMS
***                          * file named 56643082 $PPF
:END.                        * End of current component section
:DV. DV 56643082             * Get the base unmodified PPF from the DVF
***                          * file named 56643082 $PPF
:END.                        * End of current component section
:GCS. GCS 56643082 NEWNAME   * Get the base unmodified PPF from the CMS
***                          * file named 56643082 $PPF
:END.                        * End of current component section
```

Figure 29. Example of a Product Parameter Override File (VMXA $PPF)

The Temporary Product Parameter File

VMFREC, VMFAPPLY, VMFHASM, and VMFBLD all invoke the VMFOVER EXEC to create the temporary product parameter file from the product parameter file and sometimes the product parameter override file. The temporary product parameter file is a copy of the product area and one component area of the base product parameter file, with the tags from the override section or override file substituted where appropriate. If you specify the UPDATE option (the default) in the overrides, the overridden tags are included in the temporary PPF as comments. If you specify the REPLACE option, the overridden tags are not included. All the installation and service EXECs use this copy instead of the base PPF.

If you use a separate override file and specify the NEWNAME option on the *:compname.* tag in the override file the filename of the copy is the filename of the override file. Otherwise, it is the filename of the base PPF. The filetype of the copy is $PPFTEMP.

To see the temporary product parameter file before you run a service EXEC, invoke VMFOVER from the command line. You can also examine the temporary product parameter file after running the service EXECs. It is not erased.

Many different temporary product parameter files can be created from the base file on page 402, 56643082 $PPF. In the simplest example, if you issue this command:

```
vmfover 56643082 cp ■
```

you will get a temporary file just like the CP section of the base file, like the file in Figure 30 on page 421.

```
*=====================================================
*==========>   * Temporary override of CP component
*==========>   * in PPF of product 56643082
*=====================================================
*******   PRODUCT PARAMETER FILE   *******
*
*
******   Product Parameter File for
******              VMXA SP 2
*
******   NOTE: All tags must be in upper case.
*
*=========================================================================
*========== Product Header
*=========================================================================
******  List of Components in VMXA SP 2
*
:COMPLST.      CP    CMS    DV    GCS
:OVERLSTP.
:OVERLST.   CORCP CORCMS CORDV CORGCS
*
*=========================================================================
*========== End of Product Header
*=========================================================================
*********************************
:CP.   *  Parameters for CP
*********************************
*=========================================================================
*=========================================================================
*
:CNTRLOP.                    *  Section one - Control Options
*                            *   DO NOT DELETE THESE CONTROL TAGS
*=========================================================================
*
:
*=========================================================================
:MDA.                        * Section two - Mini-Disk Assignments
*
:
```

Figure 30 (Part 1 of 2). 56643082 $PPFTEMP (Without Overrides)

```
*====================================================================
:PRTFNC.                    * Section three - Part type/Function List
*
:
*
*====================================================================
:RECSER.                     * Receive (Service)
*
:
:APP.                        * Apply Function
*====================================================================
*
:
*====================================================================
:BLD.                        * Build Function
*
:
:END.                        * End of current component section
*====================================================================
```

Figure 30 (Part 2 of 2). 56643082 $PPFTEMP (Without Overrides)

If you issue:

vmfover 56643082 corcp ■

you will get a temporary file with overrides from the override section of the base file, like the file in Figure 31.

Note: When you issue VMFREC or VMFAPPLY, use a component name of CP and the COR option instead of the CORCP component name to create this temporary file.

```
*============================================
*====>   * Temporary override of CP component
*====>   * in PPF of product 56643082
*====>   * by CORCP override section
*====>   * of the PPF for the product 56643082
*============================================
*====================================================================
*            Product Parameter File for VM/XA SP Release 2
*====================================================================
*
* The following APAR's have been applied to this file:
*
*  VM32064 - 05/24/88 - New Support for Service Enhancements 2.1
*  VM32436 - 08/01/88 - New Function for Service Enhancements 2.1
*  VM35065 - 01/04/89 - Change to MACUPDT VMFRCUPD in RECSER section
*                       New load lists in CP/CMS in BLD sections
*  VM35315 - 01/06/89 - PE VM35065 Change EXECUPD VMFRCCOM to
*                       EXECUPD VMFRCUPD in RECSER section of DV and GCS
```

Figure 31 (Part 1 of 5). 56643082 $PPFTEMP (With Corrective Service Overrides)

```
*  VM35316 - 01/06/89 - Remove 19E from SYSTEM disk string
*  VM35405 - 01/09/89 - Remove DELTA2 string from GCS section in $PPF
*  VM36183 - 02/28/89 - Add CPYSYSDV to DV BLD section in $PPF. Requires
*                       Dump View CO-REQ apar VM36326.
*  VM36911 - 04/20/89 - Add HELP to DVF section of PPF. Prereqs DVF
*                       PTF which shipped this service HELP (VM36585)
*  VM37518 - 06/23/89 - New AUX file part handlers added.  New CKUPD tag
*                       added.  Reformatted for readability.  COR
*                       overrides moved to after base sections.
*
*====================================================================
*            NOTE: All tags must be in upper case.
*====================================================================
*====================================================================
* Start of Product Header - List of Components in VM/XA SP Release 2
*====================================================================
:COMPLST.      CP    CMS    DV    GCS
:OVERLSTP.
:OVERLST.
*====================================================================
* End of Product Header
*====================================================================

*====================================================================
:CP.                          * Start of Parameters for CP
*====================================================================
:CNTRLOP.                     * Section one - Control Options
**                            * DO NOT DELETE THESE CONTROL TAGS
* TAG      VALUE(S)           * DESCRIPTION
**
:CNTRL.    HCPXA              * Control File name
*====================================================================
:UPDTID. AUXCOR               * COR service applied at AUXCOR level
*:UPDTID.   AUXXA               * Update level identifier for building AUX
:SLVI.     H/HP H/PP          * System level and version indicators
:NLS.      AMENG              * System Language
:USEREXIT. VMFUECP            * User exit EXEC.  Called before initial
**                            * accesses and after final access retore.
**                            * 2 paramters passed will indicate calling
**                            * EXEC and SET-UP or CLEAN-UP.
:MERGE. LOCAL1                * Merge only LOCAL1 not APPLY and DELTA1
*:MERGE.    DELTA1 APPLY LOCAL1
**                            * Disk strings to be merged at Receive
:LTO.      NO                 * Erase lower level TEXT decks at Receive
:CKTXT.    YES                * Check for full text at Apply
:CKUPD.    NO                 * Check update fm regardless of $PTFPART fm
:CKAUX.    YES                * Check composite AUX at Build
:CKREQ.    YES                * Check Requisites at Build
*====================================================================
```

Figure 31 (Part 2 of 5). 56643082 $PPFTEMP (With Corrective Service Overrides)

```
:MDA.                          * Section two - Mini-Disk Assignments
**                             * STRING NAMES MUST START IN COLUMN 1
* STRNGNAM MINIDISKS           * DESCRIPTION
**
*TASK      191                 * Build disks accessed before rest of dbase
*LOCAL1     295 591 291         * Disks for corrective and local service
LOCAL1 295 591                 * Merge doesn't go to SUP reach-ahead disk
*LOCALn                        * You may define additional LOCAL disks
*APPLY      492 192            * Control structure built by service EXECs
APPLY  295 591 291 492 192     * LOCAL1 disks added to APPLY string
*DELTA1     594 294            * CP service from TAPE
DELTA1 295 591 291 594 294     * LOCAL1 disks added to DELTA1 string
*DELTAn                        * You may define additional DELTA disks
BASE1      194                 * CP text,loadlist,control file,maclibs
BASE2      394                 * CP source,macro defs,and copy files
*BASEn                         * You may define additional BASE disks
*BUILD1    423                 * New SYSRES for CP Nucleus Build
BUILD2     49D                 * New CMS system disk for AMENG HELP files
BUILD3     49C                 * New CMS system disk for UCENG HELP files
BUILD4     490                 * New CMS system disk for nucleus etc.
*BUILDn                        * You may define additional BUILD disks
SYSTEM     191 395 691 391 593 293 193
**                             * Disks reqd for your system environment.
**                             * CMS database needed for MACLIBs.  If
**                             * latest MACLIBs are on a build disk, only
**                             * that disk must appear here.
*============================================================================
:PRTFNC.                       * Section three - Part type/Function List
*============================================================================
:RECSER.                       * Receive (Service)
**
* TAPEFILE EXEC     TARGET     * DESCRIPTION
**
  AXLIST   VMFRCAXL DELTA1     * Apply and Exclude lists
  PARTLST  VMFRCCOM DELTA1     * PTF Parts Lists
  UPDT     VMFRCUPD DELTA1     * Update files for CP ASSEMBLE source
  TEXT     VMFRCTXT DELTA1     * Text replacement modules for CP
**                             * Text shells created on target disk
**                             * Update shells created on APPLY disk
  MACAUX   VMFRCAUX DELTA1     * AUX files for updates to non-ASSEM parts
  MACUPDT  VMFRCUPD DELTA1     * Update files for non-ASSEMBLE parts
  SOURCE   VMFRCCOM DELTA1     * New Source (assemble,macro,copy,$exec...)
  MACLIB   VMFRCCOM DELTA1     * Maclibs are not replaced on the base
  EXEC     VMFRCCOM DELTA1     * Replacement parts and regenerated
**                             * parts other than maclibs, text, modules
  HELP     VMFRCCOM BUILD2     * AMENG HELP files go directly to the
**                             * AMENG help Build disk
  HELP     VMFRCUPP BUILD3     * UCENG HELP files go directly to the
**                             * UCENG help Build disk
  RESV    *VMFRCCOM DELTA1     * Reserved parts
  CPMOD    VMFRCCOM BUILD4     * CP modules targeted for system disk
  PTFS     VMFRCCOM DELTA1     * PTF numbered parts other than TXT
```

Figure 31 (Part 3 of 5). 56643082 $PPFTEMP (With Corrective Service Overrides)

```
:APP.                          * Apply Function
**
* PARTTYPE EXEC     RENAME    * DESCRIPTION
**
  TXT*     VMFAPTXT TEXT      * Create AUX and rename appropriate update
**                             * files to fm5 for names in loadlist.
**                             * Other TXT parts renamed to TEXT
  TAM*     VMFAPNLS TXTAMENG  * For American English NLS support
  TKA*     VMFAPNLS TXTKANJI  * For KANJI NLS support
  TUC*     VMFAPNLS TXTUCENG  * For Uppercase English NLS support
  TPO*     VMFAPNLS TXTPORTG  * For Brazilian Portuguese NLS support
  TFR*     VMFAPNLS TXTFRANC  * For French NLS support
  TGR*     VMFAPNLS TXTGER    * For German NLS support
  AUXXA    VMFAPAUX           * For MACRO and EXEC AUX files
  AXA*     VMFAPAUX AUXXA     * For MACRO and EXEC AUX files
  EXEC     VMFAPCOM           *
  MACRO    VMFAPCOM           *
** .       VMFAPCOM           * ALL PARTS ARE LISTED BY FILETYPE
** .       VMFAPCOM           * If a part type is not listed
** .       VMFAPCOM           * VMFAPCOM is called by default
*====================================================================
:BLD.                          * Build Function
**
* BUILDLST EXEC     TARGET    * DESCRIPTION
**
  CPLOAD   VMFBDNUC *BUILD1   * Create CP nucleus based on loadlist
**                             * Target disk symbol agrees with
**                             * disk address in HCPSYS ASSEMBLE
**                             * on the SYSRES macro. Defaults to
**                             * what is in HCPSYS ASSEMBLE if no
**                             * target disk is specified.
**
  CPYSYSCP VMFBDCPY BUILD4    * Copy TEXT decks that do not become
**                             * part of a MODULE to the system disk
**
  CPYBTHCP VMFBDCPY DELTA1    * Copy TEXT decks used by UTILITY and
**                             * listed in the load list, CPLOAD EXEC,
**                             * to the DELTA disk. VMFAPPLY only
**                             * copies the ones not listed in the
**                             * load list.
**
* CPYUTLCP VMFBDCPY DELTA1    * Copy TEXT decks used by UTILITY which
**                             * are not listed in the load list,
**                             * CPLOAD EXEC. This list would be used
**                             * if you have local modifications and
**                             * you choose not to run VMFAPPLY. All
**                             * TEXT decks for which you have received
**                             * service and/or have local mods would
**                             * be copied as a result. You need to remove
**                             * the asterisk before CPYUTLCP before
```

Figure 31 (Part 4 of 5). 56643082 $PPFTEMP (With Corrective Service Overrides)

```
**                          * running VMFBLD.
**                          * Use UTILITY to build CP Utilities
*=========================================================================
:END.                       * End of Parameters for CP
```

Figure 31 (Part 5 of 5). 56643082 $PPFTEMP (With Corrective Service Overrides)

If you issue:

vmfover vmxa cp ■

you will get a temporary file, like the file in Figure 32, with overrides from the alias override file, VMXA $PPF.

```
*=========================================
*====>   * Temporary override of CP component
*====>   * in PPF of product 56643082
*====>   * by CP override section
*====>   * of the PPF for the product 56643082
*=========================================
*=========================================================================
*            Product Parameter File for VM/XA SP Release 2
*=========================================================================
*
* The following APAR's have been applied to this file:
*
*  VM32064 - 05/24/88 - New Support for Service Enhancements 2.1
*  VM32436 - 08/01/88 - New Function for Service Enhancements 2.1
*  VM35065 - 01/04/89 - Change to MACUPDT VMFRCUPD in RECSER section
*                       New load lists in CP/CMS in BLD sections
*  VM35315 - 01/06/89 - PE VM35065 Change EXECUPD VMFRCCOM to
*                       EXECUPD VMFRCUPD in RECSER section of DV and GCS
*  VM35316 - 01/06/89 - Remove 19E from SYSTEM disk string
*  VM35405 - 01/09/89 - Remove DELTA2 string from GCS section in $PPF
*  VM36183 - 02/28/89 - Add CPYSYSDV to DV BLD section in $PPF. Requires
*                       Dump View CO-REQ apar VM36326.
*  VM36911 - 04/20/89 - Add HELP to DVF section of PPF. Prereqs DVF
*                       PTF which shipped this service HELP (VM36585)
*  VM37518 - 06/23/89 - New AUX file part handlers added.  New CKUPD tag
*                       added.  Reformatted for readability.  COR
*                       overrides moved to after base sections.
*
*=========================================================================
*            NOTE: All tags must be in upper case.
*=========================================================================

*=========================================================================
* Start of Product Header - List of Components in VM/XA SP Release 2
*=========================================================================
:COMPLST.      CP    CMS    DV    GCS
:OVERLSTP.
```

Figure 32 (Part 1 of 4). 56643082 $PPFTEMP (with Alias Overrides)

```
:OVERLST.
*====================================================================
* End of Product Header
*====================================================================

*====================================================================
:CP.                        * Start of Parameters for CP
*====================================================================
:CNTRLOP.                   * Section one - Control Options
**                          * DO NOT DELETE THESE CONTROL TAGS
* TAG       VALUE(S)         * DESCRIPTION
**
:CNTRL.     HCPXA            * Control File name
:UPDTID.    AUXXA            * Update level identifier for building AUX
:SLVI.      H/HP H/PP        * System level and version indicators
:NLS.       AMENG            * System Language
:USEREXIT.  VMFUECP          * User exit EXEC.  Called before initial
**                          * accesses and after final access retore.
**                          * 2 paramters passed will indicate calling
**                          * EXEC and SET-UP or CLEAN-UP.
:MERGE.     DELTA1 APPLY LOCAL1
**                          * Disk strings to be merged at Receive
:LTO.       NO               * Erase lower level TEXT decks at Receive
:CKTXT.     YES              * Check for full text at Apply
:CKUPD.     NO               * Check update fm regardless of $PTFPART fm
:CKAUX.     YES              * Check composite AUX at Build
:CKREQ.     YES              * Check Requisites at Build
*====================================================================
:MDA.                       * Section two - Mini-Disk Assignments
**                          * STRING NAMES MUST START IN COLUMN 1
* STRNGNAM MINIDISKS         * DESCRIPTION
**
*TASK       191              * Build disks accessed before rest of dbase
LOCAL1      295 591 291      * Disks for corrective and local service
*LOCALn                     * You may define additional LOCAL disks
APPLY       492 192          * Control structure built by service EXECs
DELTA1      594 294          * CP service from TAPE
*DELTAn                     * You may define additional DELTA disks
BASE1       194              * CP text,loadlist,control file,maclibs
BASE2       394              * CP source,macro defs,and copy files
*BASEn                      * You may define additional BASE disks
*BUILD1     423              * New SYSRES for CP Nucleus Build
BUILD2      49D              * New CMS system disk for AMENG HELP files
BUILD3      49C              * New CMS system disk for UCENG HELP files
BUILD4      490              * New CMS system disk for nucleus etc.
*BUILDn                     * You may define additional BUILD disks
SYSTEM      191 395 691 391 593 293 193
**                          * Disks reqd for your system environment.
**                          * CMS database needed for MACLIBs.  If
**                          * latest MACLIBs are on a build disk, only
**                          * that disk must appear here.
```

Figure 32 (Part 2 of 4). 56643082 $PPFTEMP (with Alias Overrides)

```
*====================================================================
:PRTFNC.                        * Section three - Part type/Function List
*====================================================================
:RECSER.                        * Receive (Service)
**
* TAPEFILE EXEC     TARGET     * DESCRIPTION
**
  AXLIST   VMFRCAXL DELTA1     * Apply and Exclude lists
  PARTLST  VMFRCCOM DELTA1     * PTF Parts Lists
  UPDT     VMFRCUPD DELTA1     * Update files for CP ASSEMBLE source
  TEXT     VMFRCTXT DELTA1     * Text replacement modules for CP
**                             * Text shells created on target disk
**                             * Update shells created on APPLY disk
  MACAUX   VMFRCAUX DELTA1     * AUX files for updates to non-ASSEM parts
  MACUPDT  VMFRCUPD DELTA1     * Update files for non-ASSEMBLE parts
  SOURCE   VMFRCCOM DELTA1     * New Source (assemble,macro,copy,$exec...)
  MACLIB   VMFRCCOM DELTA1     * Maclibs are not replaced on the base
  EXEC     VMFRCCOM DELTA1     * Replacement parts and regenerated
**                             * parts other than maclibs, text, modules
  HELP     VMFRCCOM BUILD2     * AMENG HELP files go directly to the
**                             * AMENG help Build disk
  HELP     VMFRCUPP BUILD3     * UCENG HELP files go directly to the
**                             * UCENG help Build disk
  RESV    *VMFRCCOM DELTA1     * Reserved parts
  CPMOD    VMFRCCOM BUILD4     * CP modules targeted for system disk
  PTFS     VMFRCCOM DELTA1     * PTF numbered parts other than TXT
*====================================================================
:APP.                           * Apply Function
**
* PARTTYPE EXEC     RENAME     * DESCRIPTION
**
  TXT*     VMFAPTXT TEXT       * Create AUX and rename appropriate update
**                             * files to fm5 for names in loadlist.
**                             * Other TXT parts renamed to TEXT
  TAM*     VMFAPNLS TXTAMENG   * For American English NLS support
  TKA*     VMFAPNLS TXTKANJI   * For KANJI NLS support
  TUC*     VMFAPNLS TXTUCENG   * For Uppercase English NLS support
  TPO*     VMFAPNLS TXTPORTG   * For Brazilian Portuguese NLS support
  TFR*     VMFAPNLS TXTFRANC   * For French NLS support
  TGR*     VMFAPNLS TXTGER     * For German NLS support
  AUXXA    VMFAPAUX            * For MACRO and EXEC AUX files
  AXA*     VMFAPAUX AUXXA      * For MACRO and EXEC AUX files
  EXEC     VMFAPCOM            *
  MACRO    VMFAPCOM            *
** .       VMFAPCOM            * ALL PARTS ARE LISTED BY FILETYPE
** .       VMFAPCOM            * If a part type is not listed
** .       VMFAPCOM            * VMFAPCOM is called by default
```

Figure 32 (Part 3 of 4). 56643082 $PPFTEMP (with Alias Overrides)

```
*====================================================================
:BLD.                          * Build Function
**
* BUILDLST EXEC     TARGET    * DESCRIPTION
**
  CPLOAD   VMFBDNUC *BUILD1   * Create CP nucleus based on loadlist
**                             * Target disk symbol agrees with
**                             * disk address in HCPSYS ASSEMBLE
**                             * on the SYSRES macro. Defaults to
**                             * what is in HCPSYS ASSEMBLE if no
**                             * target disk is specified.
**
  CPYSYSCP VMFBDCPY BUILD4    * Copy TEXT decks that do not become
**                             * part of a MODULE to the system disk
**
  CPYBTHCP VMFBDCPY DELTA1    * Copy TEXT decks used by UTILITY and
**                             * listed in the load list, CPLOAD EXEC,
**                             * to the DELTA disk. VMFAPPLY only
**                             * copies the ones not listed in the
**                             * load list.
**
* CPYUTLCP VMFBDCPY DELTA1    * Copy TEXT decks used by UTILITY which
**                             * are not listed in the load list,
**                             * CPLOAD EXEC. This list would be used
**                             * if you have local modifications and
**                             * you choose not to run VMFAPPLY. All
**                             * TEXT decks for which you have received
**                             * service and/or have local mods would
**                             * be copied as a result. You need to remove
**                             * the asterisk before CPYUTLCP before
**                             * running VMFBLD.
**                             * Use UTILITY to build CP Utilities
*====================================================================
:END.                          * End of Parameters for CP
```

Figure 32 (Part 4 of 4). 56643082 $PPFTEMP (with Alias Overrides)

The Program Update Tape (PUT)

The format of each volume of the PUT is as follows:

- **The first tape file** contains the PUT DOCUMENT, the VMSERV EXEC, and the latest tested level of the service EXECs (with PTF-numbered filenames).
- **The second tape file** contains a Memo to Users for each program product service on the volume, and the latest tested level of the product parameter file for each product on the tape.
- **The remaining tape files** contain service files for the program products on that volume.

Figure 33 shows the format of the program update tape.

```
------------------------------------------------------------------
        --- Program Update Tape (PUT) Format for VM/XA SP 2 ---
------------------------------------------------------------------


------------------------------------------------------------------
|                        --- Tape File 1 ---                      |
------------------------------------------------------------------
PUT       8801             <=== PUT Descriptor File
PUT       Document         <=== PUT Content and Instructions
VMSERV    EXEC             <=== Compatible Service EXEC
VMFREC    EXCnnnnn         <=== Latest tested level of service EXECs
VMFBLD    EXCnnnnn
VMFAPPLY  EXCnnnnn
:
$LEVEL    MAP              <=== Inventory of service EXECs
    ========================= Tape Mark =========================

------------------------------------------------------------------
|                        --- Tape File 2 ---                      |
------------------------------------------------------------------
prodid1   crllnn1          <=== Program Level File
prodid1   MEMO             <=== Instructions for product
prodid    $PFnnnnn         <=== Product Parameter File
prodid2   crllnn0
prodid2   MEMO
prodid3   crllnn1
prodid3   MEMO
prodid    $PFnnnnn         <=== Product Parameter File
prodid4   crllnn1
prodid4   MEMO
prodid    $PFnnnnn         <=== Product Parameter File
:
    ========================= Tape Mark =========================
```

Figure 33 (Part 1 of 3). Format of the Program Update Tape

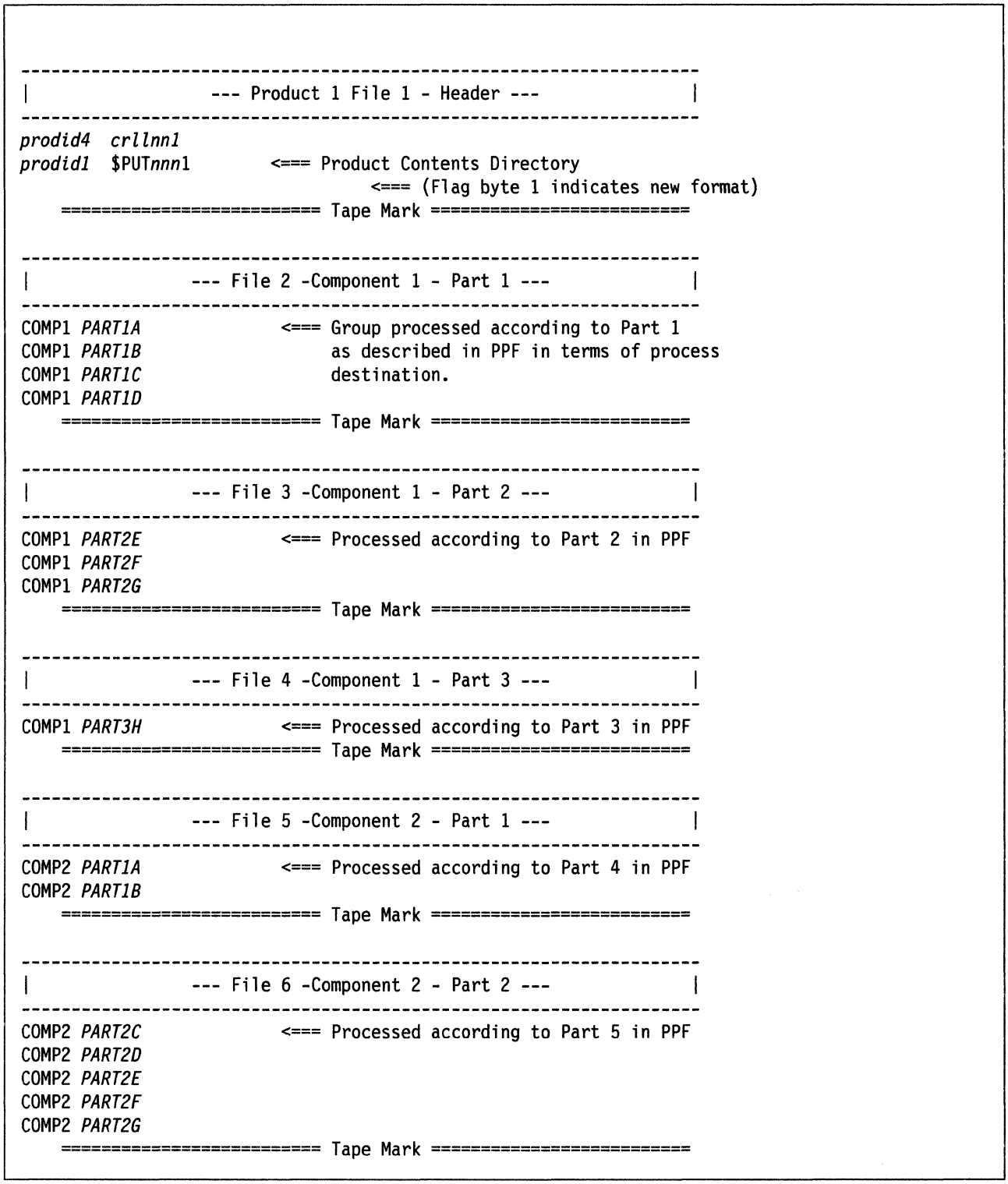


Figure 33 (Part 2 of 3). Format of the Program Update Tape

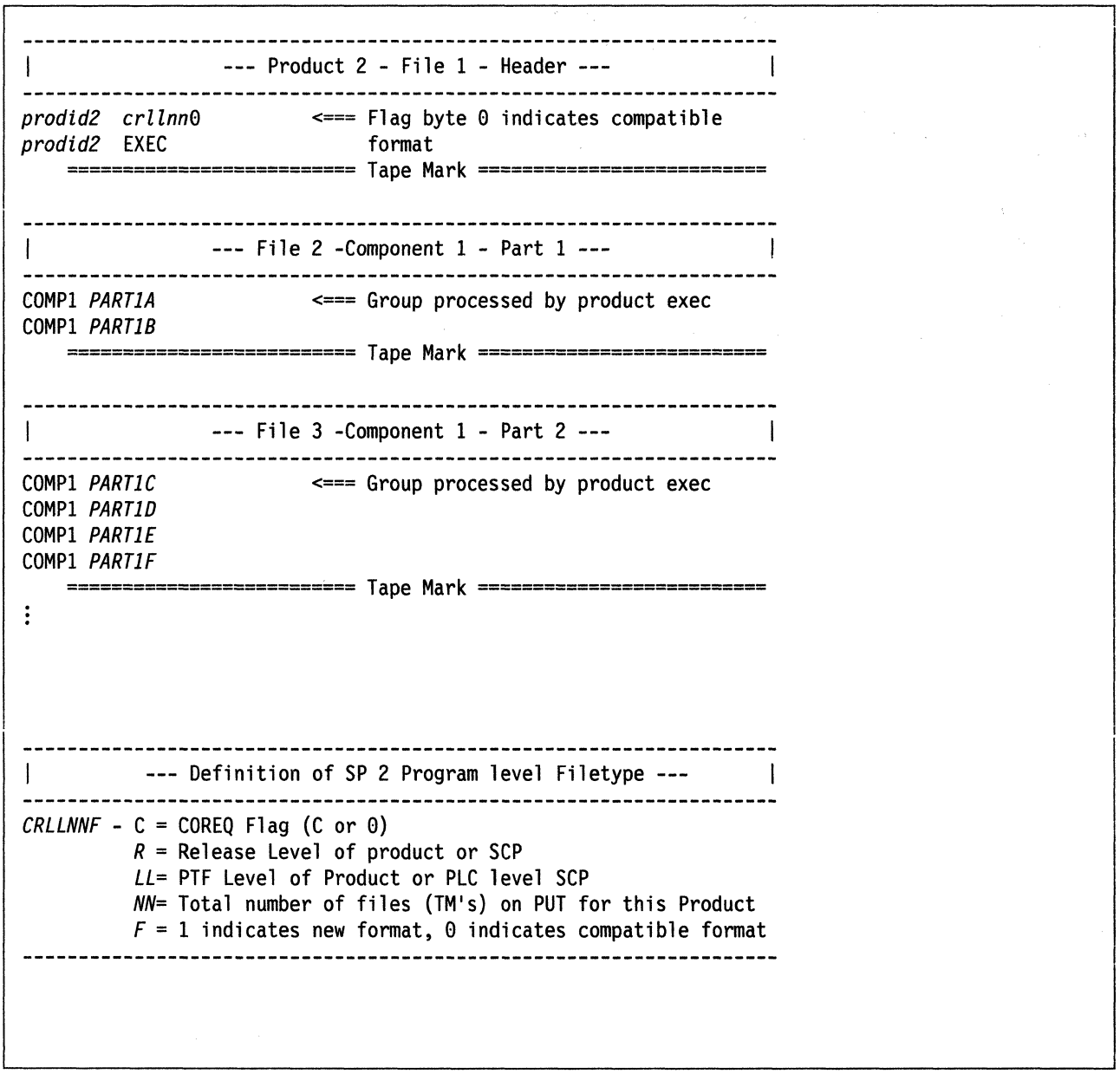


Figure 33 (Part 3 of 3). Format of the Program Update Tape

Although the VMSERV EXEC and service EXECs for each product are still provided on the PUT tape, you must use the new service EXECs, VMFREC, VMFAPPLY, and VMFBLD, to receive and apply service to CP, CMS, GCS, and the dump viewing facility. These EXECs are loaded from the VM/XA SP product tape when you install the system. You can still use VMSERV to receive and apply service to other program products. See Chapter 9 through Chapter 17 for examples of using the new EXECs.

IBM packages service for VM/XA SP as service level updates. Periodically, or whenever you request it, IBM ships a service level update. This includes cumulative preventive maintenance since the last release of VM/XA SP.

You may find that there is more than one preventive update on a program update tape.

The specific steps for applying the service for each particular service level update are in the PUT documentation. Chapter 9, "Receiving Program Update Service or Corrective Service for CMS," Chapter 10, "Receiving Program Update Service or Corrective Service for CP," Step 2 of Chapter 15,

"Program Update Service or Corrective Service to the Dump Viewing Facility," and Step 2 of Chapter 16, "Program Update Service or Corrective Service to GCS" tell you how to get this documentation.

The Corrective Service Tape (COR)

The format of each volume of the COR is as follows:

- **The first tape file** contains the COR DOCUMENT and the latest tested level of the service EXECs (with PTF-numbered filenames).
- **The second tape file** contains a Memo to Users for each program product service on the volume, and the latest tested level of the product parameter file for each product on the tape.
- **The remaining tape files** contain service files for the program products on that volume.

For new format products, the format of the corrective service tape is exactly the same as the format of the program update tape, except that only the tape files containing parts that actually need corrective service are included.

Chapter 9. Receiving Program Update Service or Corrective Service for CMS

This chapter describes:

- How to receive service from the program update tape or corrective service tape for CMS.

Step 1. Preparation

Before you begin processing program update service or corrective service, you must:

1. Establish the appropriate minidisk access order.

   Examine the SETUP EXEC, which accesses all the minidisks where the service EXECs and the product parameter file might be found. The SETUP EXEC reflects the minidisk access order in the IBM-supplied product parameter file. If you have changed the access order, copy the SETUP EXEC with another name and edit your new EXEC to reflect your own product parameter file.

   The SETUP EXEC does not access all the minidisks needed by the service EXECs; and your new EXEC, if you create one, does not have to do so. Each service EXEC will invoke VMFSETUP to create the access order it needs.

   The access order created by the SETUP EXEC is:

Mode	Address	Function
A	191	MAINT's work disk
B	395	CMS alternate LOCAL1 disk
E	691	CMS intermediate alternate LOCAL1 disk
F	391	CMS current LOCAL1 disk
G	692	CMS alternate APPLY disk
H	392	CMS current APPLY disk
I	593	CMS alternate DELTA disk
J	293	CMS current DELTA disk
K	193	CMS current BASE1 disk
L	393	CMS current BASE2 disk
M	490	CMS alternate BUILD1 disk
Note: No C- or D-disk is accessed. If you have a C-disk accessed, release it. You do not need to release your D-disk.		

Now run the SETUP EXEC or your equivalent EXEC:

```
setup ■
```

2. Copy the latest version of the DMSNGP ASSEMBLE file to the 395 disk. (If there is already a copy of DMSNGP ASSEMBLE or DMSNGP TEXT on the 395 disk, rename it first.) Edit the copy and make the following changes:
   a. Change SYSDISK=190 to SYSDISK=490.
   b. Change IPLADDR=190 to IPLADDR=490.
   c. Change HELP=19D to HELP=49D (or HELP=19C to HELP=49C).
   d. Change SYSNAME=CMS to SYSNAME=*newname*, where *newname* is the name you choose for your test system. The sample procedure in Chapter 13, "Rebuilding CMS after Applying Service," uses ALTCMS.
   e. Change INSTSEG=CMSINST to INSTSEG=*newname*, where *newname* is the name you choose for your test installation segment. The sample procedure in Chapter 13, "Rebuilding CMS after Applying Service," uses ALTINST.
   f. Change the VERSION= and INSTID= parameters to identify your test system.
   g. Make sure that SAVESYS=NO. If SAVESYS=YES, when you IPL CMS the system will try to save the alternate CMS segment that you will define in Step 10 of Chapter 13, "Rebuilding CMS after Applying Service" in the existing CMS segment.

   These changes will cause the new CMS nucleus (with service) to be built on the 490 minidisk. (If you have applied service before and the nucleus you are using now is on the 490 minidisk, SYSDISK and IPLADDR are already 490. Change them back to 190 so that the new nucleus will be built on the 190 minidisk.)
3. You will probably want VMFREC to merge the alternate DELTA, APPLY, and LOCAL minidisks to their corresponding current disks, in accordance with the :MERGE. tag in the product parameter file. If for some reason you want to do it manually, or if you want to keep any disk string from being merged, see Appendix G, "Controlling Disk String Merges" on page 859.

   Note: When you apply corrective service, the DELTA1 and APPLY disk strings will be redefined to be the same as the LOCAL1 disk string in the temporary product parameter file. This redefinition is necessary in order for VMFREC to load files to the LOCAL1 disk. It is done automatically by the CORCMS override in the product parameter file.
4. Use the DASD DUMP Restore program to copy the current BUILD1 disk to its corresponding alternate disk.

```
ddr ■
VM/XA SYSTEM PRODUCT DASD DUMP RESTORE PROGRAM
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS.
ENTER:
```

```
sysprint cons ■
ENTER:
```

This command tells DDR to send program messages to your console.

```
input 190 devtype MNT190 ■
ENTER:
```

190 is your current CMS system disk. *devtype* is the device type of the DASD volume where 190 is located.

```
output 490 devtype MNT490 ■
ENTER:
```

490 is your alternate CMS system disk. *devtype* is the device type of the DASD volume where 490 is located.

```
copy 000 endcyl ■
```

The value for *endcyl* depends on the device type of your 190 disk:

Device Type	*endcyl*
3350	73
3375	112
3380	71

```
DMKDDR711R Volid Read is MNT190.  Do you
           wish to continue?
yes ■
ENTER NEXT EXTENT OR NULL LINE:
■
⋮
END OF COPY
ENTER:
■
END OF JOB
```

You have not yet changed the label of the 490 minidisk. (You will do so in substep 5.)

5. Relabel the 490 minidisk:

```
access 490 t ■
Ready; T=n.nn/n.nn hh:mm:ss
format 490 t (label ■
ENTER LABEL:
mnt490 ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Access 490 *after* the system disk. You *must* use the **label** option. If you don't, you will erase all the files on the 490 minidisk.

6. Repeat substeps 4 on page 436 and 5 to copy the current HELP disks (19D, 19C, 19B...) to their corresponding alternate disks (49D, etc.) and to relabel the alternate disks as MNT49*n*. Check the minidisk definitions in your user directory to find the appropriate value for *endcyl*.
7. Determine your system default national language:

```
query lang ■
langid
```

langid can be:

Langid	Language
AMENG	Mixed-case American English
KANJI	Kanji (Japan)
UCENG	Uppercase American English
PORTG	Brazilian Portuguese
FRANC	French
GER	German

If the system default national language is uppercase American English (UCENG), you must make these changes:

a. Change the :NLS. tags in the product parameter file from AMENG to UCENG.

b. Convert the following files to upper case:

   - $VMFMSG$ EXEC
   - $VMFMSG$ $EXEC

```
xedit $vmfmsg$ {exec|$exec} ■
uppercas * ■
file ■
```

   Note: If you receive service to these files, you must copy the changes in the replacement files to your modified version.

c. In the DMSNGP ASSEMBLE file, change the LANG parameter from AMENG to UCENG and the HELP disk address from 19D to 19C.

8. If your product parameter file is on the 191 disk, you can either rename it or copy it to another disk so that VMFREC will not overlay it, or follow the instructions in the "Usage Notes" section of the VMFREC EXEC description (on page 710) for instructions on how to set up a separate minidisk for the service execs.

9. Rename the latest level of DMSNGP TEXT to save your old text file, then reassemble the DMSNGP ASSEMBLE file to include the changes you made in steps 2 on page 436 and 7 on page 437.

```
vmfhasm dmsngp 56643082 cms ■
```

Step 2. Receive Service

Before You Receive Service

If you are not familiar with the new options provided in the VMFREC EXEC with the addition of APAR VM37518, refer to "VMFREC EXEC" on page 709. In particular, the section "Usage Notes" on page 710 describes considerations for using the TASK disk string to establish a separate disk string for the service execs.

1. Establish the appropriate minidisk access order:

   setup ■

2. Attach a tape drive to MAINT:

   attach *vdevno* *** 181** ■

3. Mount and ready the program update tape or corrective service tape.

4. Map the PUT or COR tape and receive the documentation. Mapping the tape means verifying that all the necessary files exist. Enter:

   release c ■

   VMFREC INFO (MEMOS xxx loads documents to the C-disk if it is accessed, otherwise to the A-disk. You want them loaded to the A-disk.

   vmfrec info (MEMOS *xxx* ■
   DMSREC1852I This is *n* of *n*,
   level *level* {PUT|COR} tape

   xxx is **put** if you are mapping a program update tape, **cor** if you are mapping a corrective service tape. If the tape you mounted is not the kind of tape you specified, you receive an error message.

   VMFREC INFO (MEMOS maps the tape and loads the tape map, the PUT document or COR document, and the Memo to Users to your A-disk. Read these documents before going on.

   If your system default language is uppercase American English, you may get message DMSMGM814E. This message is caused by a problem with the REXX date function. You can ignore it.

   Ready; T=*n.nn/n.nn hh:mm:ss*

5. VMFREC may tell you that the level of service EXECs on the tape is different than the level of service EXECs on your system. Refer to the "Usage Notes" section of the VMFREC EXEC description (on page 710) for information on receiving a new level of the service EXECs.

   - If you have not made any local modifications to the old product parameter file, you can use the new file as it stands. Go on to the next step.

- If you had any local modifications in the old product parameter file, your override file should be on the CMS LOCAL1 disk string. Look at the new product parameter file on the 191 disk to see if any changes from the last IBM-supplied product parameter file affect your override file. If you are applying corrective service, be sure to include the changes from the corrective service override section of the base file in your override file.

  The product parameter file has three sections for each component:

  a. Control options
  b. Minidisk assignments
  c. EXECs needed to update each part of the product.

  You may change the first two sections to suit your installation, but it is not recommended that you change the EXEC section.

  Always place changes in an override file. Do not change the base product parameter file.

  Warning: The disk addresses in the SPLOAD PROFILE, user directory, 56643082 $PPF, and SETUP EXEC, and the default addresses used by the ITASK EXEC must all match. If you change an address in any one of these places, you must change it in all the others. We recommend that you do not change any addresses.

  After you have made any necessary changes in your own override file, copy it to the 191 disk with the REPLACE option.

For more information about the product parameter file, see "The Product Parameter File" on page 401.

6. Receive the service files to the target minidisk you specified in the product parameter file. Enter:

```
vmfrec 56643082 cms (xxx ■
DMSREC1852I This is n of n,
            level level {PUT|COR} tape
```

xxx is **put** for program update service, **cor** for corrective service. If you need to invoke an override file, substitute that file's name for **cms**.

```
DMSREC1869R 56643082 CMS begins on {PUT|COR}
             level volume vol.  Mount
             correct volume and press ENTER
             or type QUIT.
■
DMSREC1852I This is n of n,
            level level {PUT|COR} tape
DMSREC1804I Receiving service for
            component CMS of product
            56643082
DMSREC1851I Processing parttype with the part handler
            parthandler EXEC.
⋮
DMSREC1850I The processing for RESV by the
            VMFRCCOM EXEC was bypassed
Ready; T=n.nn/n.nn hh:mm:ss
```

If you do not have the correct volume mounted, mount it now.

The VMFREC EXEC loads the service files to the target minidisk.

7. Review the receive exception log ($VMFREC $ERRLOG). If necessary, correct any problems before going on.

```
vmfview receive ■
Ready; T=n.nn/n.nn hh:mm:ss
```

MAP 0006: What to Do Next

001

Are you also servicing CP?

Yes **No**

002

– Go to Chapter 11, "Applying Program Update Service or Corrective Service to CMS" on page 449.

003

Have you received service for CP?

Yes **No**

004

– Go to Chapter 10, "Receiving Program Update Service or Corrective Service for CP" on page 443.

005

– Go to Chapter 11, "Applying Program Update Service or Corrective Service to CMS" on page 449.

Chapter 10. Receiving Program Update Service or Corrective Service for CP

This chapter describes:

- How to receive service from the program update tape or corrective service tape for CP.

Step 1. Preparation

Before you begin processing program update service or corrective service, you must:

1. Establish the appropriate minidisk access order. If you have not yet examined the SETUP EXEC, and created your own version if necessary, turn to page 435 and do so now. Then issue:

   ```
   setup ■
   ```

2. You will probably want VMFREC to merge the alternate DELTA, APPLY, and LOCAL minidisks to their corresponding current disks, in accordance with the :MERGE. tag in the product parameter file. If for some reason you want to do it manually, or if you want to keep any disk string from being merged, see Appendix G, "Controlling Disk String Merges" on page 859.

   Note: When you apply corrective service, the DELTA1 and APPLY disk strings will be redefined to be the same as the LOCAL1 disk string in the temporary product parameter file. This redefinition is necessary in order for VMFREC to load files to the LOCAL1 disk. It is done automatically by the CORCP override in the product parameter file.

3. Determine your system default national language:

   ```
   query lang ■
   langid
   ```

   langid can be:

Langid	Language
AMENG	Mixed-case American English
KANJI	Kanji (Japan)
UCENG	Uppercase American English
PORTG	Brazilian Portuguese
FRANC	French
GER	German

   If the system default national language is uppercase American English (UCENG), you must make these changes:

   a. Change the :NLS. tags in the product parameter file from AMENG to UCENG.

   b. Convert the following files to upper case:

      - $VMFMSG$ EXEC
      - $VMFMSG$ $EXEC

   ```
   xedit $vmfmsg$ {EXEC|$EXEC} ■
   uppercas * ■
   file ■
   ```

Note: If you receive service to these files, you must copy the changes in the replacement files to your modified version.

4. If your product parameter file is on the 191 disk, you can rename it or copy it to another disk so that VMFREC will not overlay it, or you can follow the instructions in the "Usage Notes" section of the VMFREC EXEC description (on page 710) for instructions on how to set up a separate minidisk for the service EXECs.

Step 2. Receive Service

Before You Receive Service

If you are not familiar with the new options provided in the VMFREC EXEC with the addition of APAR VM37518, refer to "VMFREC EXEC" on page 709. In particular, the section "Usage Notes" on page 710 describes considerations for using the TASK disk string to establish a separate disk string for the service EXECs.

1. Establish the appropriate minidisk access order:

   setup ■

2. Attach a tape drive to MAINT:

   attach *vdevno* *** 181** ■

3. Mount and ready the program update tape or corrective service tape.

4. Map the PUT or COR tape and receive the documentation. Mapping the tape means verifying that all the necessary files exist. Enter:

   release c ■

   VMFREC INFO (MEMOS *xxx* loads documents to the C-disk if it is accessed, otherwise to the A-disk. You want them loaded to the A-disk.

   vmfrec info (memos *xxx* ■
   DMSREC1852I This is *n* of *n*,
   level *level* {PUT|COR} tape

   xxx is **put** if you are mapping a program update tape, **cor** if you are mapping a corrective service tape. If the tape you mounted is not the kind of tape you specified, you receive an error message.

   VMFREC INFO (MEMOS maps the tape and loads the tape map, the PUT document or COR document, and the Memo to Users to your A-disk. Read these documents before going on.

   If your system default language is uppercase American English, you may get message DMSMGM814E. This message is caused by a problem with the REXX date function. You can ignore it.

   Ready; T=*n.nn/n.nn hh:mm:ss*

5. VMFREC may tell you that the level of service EXECs on the tape is different than the level of service EXECs on your system. Refer to the "Usage Notes" section of the VMFREC EXEC description (on page 710) for information on receiving a new level of the service EXECs.

   - If you have not made any local modifications to the old product parameter file, you can use the new file as it stands. Go on to the next step.

- If you have any local modifications in the old product parameter file, your override file should be on the CMS LOCAL1 disk string. Look at the new product parameter file on the 191 disk to see if any changes from the last IBM-supplied product parameter file affect your override file. If you are applying corrective service, be sure to include the changes from the corrective service override section of the base file in your override file.

  The product parameter file has three sections for each component:

  a. Control options
  b. Minidisk assignments
  c. EXECs needed to update each part of the product.

  You may change the first two sections to suit your installation, but it is not recommended that you change the EXEC section.

  Always place changes in an override file. Do not change the base product parameter file.

  Warning: The disk addresses in the SPLOAD PROFILE, user directory, 56643082 $PPF, and SETUP EXEC, and the default addresses used by the ITASK EXEC must all match. If you change an address in any one of these places, you must change it in all the others. We recommend that you do not change any addresses.

  After you have made any necessary changes in your own override file, copy it to the 191 disk with the REPLACE option.

For more information about the product parameter file, see "The Product Parameter File" on page 401.

6. Receive the service files to the target minidisk you specified in the product parameter file. Enter:

```
vmfrec 56643082 cp (xxx ■
DMSREC1852I This is n of n,
            level level {PUT|COR} tape


DMSREC1869R 56643082 CP begins on {PUT|COR}
             level volume vol.  Mount
             correct volume and press ENTER
             or type QUIT.
■
DMSREC1852I This is n of n,
            level level {PUT|COR} tape
DMSREC1804I Receiving service for
             component CP of product
             56643082
DMSREC1851I Processing parttype with part handler
             parthandler EXEC.
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

xxx is **put** for program update service, **cor** for corrective service. If you need to invoke an override file, substitute that file's name for **cp**.

If you do not have the correct volume mounted, mount it now.

The VMFREC EXEC loads the service files to the target minidisk.

7. Review the receive exception log ($VMFREC $ERRLOG). If necessary, correct any problems before going on.

```
vmfview receive ■
Ready; T=n.nn/n.nn hh:mm:ss
```

MAP 0007: What to Do Next

001

Are you also servicing CMS?

Yes **No**

002

– Go to Chapter 12, "Applying Program Update Service or Corrective Service to CP" on page 453.

003

Have you received service for CMS?

Yes **No**

004

– Go to Chapter 9, "Receiving Program Update Service or Corrective Service for CMS" on page 435.

005

– Go to Chapter 11, "Applying Program Update Service or Corrective Service to CMS" on page 449.

Chapter 11. Applying Program Update Service or Corrective Service to CMS

This chapter describes:

- How to apply service from the program update tape or corrective service tape to CMS.

Warning: If you are servicing both CP and CMS, you **must** receive both CP and CMS service before applying service to CMS, because you need CP macro libraries to apply service to CMS. Some of these libraries may have been serviced.

MAP 0008: Should You Be Doing This Now?

001

Are you also servicing CP?
Yes **No**

002

– Continue with this chapter.

003

Have you run VMFREC to receive service for CP?
Yes **No**

004

– Go to Chapter 10, "Receiving Program Update Service or Corrective Service for CP" on page 443.

005

– Continue with this chapter.

1. Consult your IBM service representative for the appropriate preventive service planning (PSP) bucket for your system's VMSUP level and PUT level. Check the bucket and add the appropriate entries to the exclude list. (The exclude list is called *fn* $EXCLIST, where *fn* is the filename of the control file defined for CMS in the product parameter file.)

   Note: If two PTFs are in the same text deck, you cannot exclude the first unless you also exclude the second, even if you list the first one on the exclude list. This restriction applies because the first PTF is considered to be a prereqisite for the second PTF.

   If you list the first PTF in the exclude list, but not the second, and if the second PTF appears in the apply list, both PTFs will be applied. You will receive an error message indicating that a PTF you wanted to exclude has been applied.

2. Establish the appropriate minidisk access order:

   setup ■

3. Apply the service, that is, create auxiliary control files. Enter:

   vmfapply 56643082 cms (*xxx* ■
   DMSAPP1853I Processing PTF *ptfnum*
   ⋮
   Ready; T=*n.nn/n.nn hh:mm:ss*

   xxx is **put** if you are applying program update service, **cor** if you are applying corrective service.

   The VMFAPPLY EXEC creates or edits the necessary control files and, if appropriate, calls the VMFAPNLS EXEC to create auxiliary control files for national language support (see "VMFAPNLS EXEC" on page 676).

4. Review the apply exception log ($VMFAPP $ERRLOG). If necessary, correct any problems before going on.

   vmfview apply ■

   If you receive message number DMSAPP1885W, VMFAPPLY has applied service to a module for which you have a local modification. Decide what you want to do with the local modification:

   - To remove a local modification, see Chapter 18, "Receiving and Applying Local Service" on page 565, and Chapter 20, "Removing Service from VM/XA SP" on page 581.
   - To rework a local modification, see Chapter 18, "Receiving and Applying Local Service" on page 565.
   - To keep a local modification, do nothing now. You will reassemble the module to pick up the local modification in Step 4 of Chapter 13, "Rebuilding CMS after Applying Service."

   Whether you remove, rework, or keep local modifications, you do not have to reissue VMFAPPLY.

   Do not erase $VMFAPP $ERRLOG.

5. Check to see if you have applied service to DMSZIT ASSEMBLE:

   filelist dmszit * *fm*5 ■

   The updates are on the alternate LOCAL1 (395) disk if you are applying corrective service, the alternate DELTA (593) disk if you are applying program update service.

   A filemode of 5 indicates that service has been applied.

If any update files are listed, or if you know that you have applied a local update to DMSZIT ASSEMBLE, copy the latest level of the text deck to the 395 disk, giving the copy the filetype TEXT:

```
copy dmszit txtxxxxx fm = text b ■
```

The text deck is on the alternate LOCAL1 (395) disk if you are applying corrective service, the alternate DELTA (593) disk if you are applying program update service.

MAP 0009: What to Do Next

001

Are you also servicing CP?

Yes **No**

002

– Go to Chapter 13, "Rebuilding CMS after Applying Service" on page 457.

003

– Go to Chapter 12, "Applying Program Update Service or Corrective Service to CP" on page 453.

Chapter 12. Applying Program Update Service or Corrective Service to CP

This chapter describes:

- How to apply service from the program update tape or corrective service tape to CP.

MAP 0010: Should You Be Doing This Now?

001

Are you also servicing CMS?
Yes **No**

002

– Continue with this chapter.

003

Have you run VMFAPPLY to apply service to CMS?
Yes **No**

004

– Go to Chapter 11, "Applying Program Update Service or Corrective Service to CMS" on page 449.

005

– Continue with this chapter.

1. Consult your IBM service representative for the appropriate preventive service planning (PSP) bucket for your system's VMSUP level and PUT level. Check the bucket and add the appropriate entries to the exclude list. (The exclude list is called *fn* $EXCLIST, where *fn* is the filename of the control file defined for CP in the product parameter file.)

   Note: If two PTFs are in the same text deck, you cannot exclude the first unless you also exclude the second, even if you list the first one on the exclude list. This restriction applies because the first PTF is considered to be a prereqisite for the second PTF.

   If you list the first PTF in the exclude list, but not the second, and if the second PTF appears in the apply list, both PTFs will be applied. You will receive an error message indicating that a PTF you wanted to exclude has been applied.

2. Establish the proper minidisk access order:

   ```
   setup ■
   ```

3. Apply the service, that is, create auxiliary control files. Enter:

   ```
   vmfapply 56643082 cp (xxx ■
   DMSAPP1853I Processing PTF ptfnum
   ⋮
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   xxx is **put** for program update service, **cor** for corrective service.

   The VMFAPPLY EXEC creates or edits the necessary control files and calls the appropriate part-processing EXECs: VMFAPTXT, VMFAPNLS, etc.

4. Review the apply exception log ($VMFAPP $ERRLOG). If necessary, correct any problems before going on.

   ```
   vmfview apply ■
   ```

   If you receive message number DMSAPP1885W, VMFAPPLY has applied service to a module for which you have a local modification. Decide what you want to do with the local modification:

   - To remove a local modification, see Chapter 18, "Receiving and Applying Local Service" on page 565, and Chapter 20, "Removing Service from VM/XA SP" on page 581.
   - To rework a local modification, see Chapter 18, "Receiving and Applying Local Service" on page 565.
   - To keep a local modification, do nothing now. You will reassemble the module to pick up the local modification in Step 4 of Chapter 14, "Rebuilding CP after Applying Service."

   Whether you remove, rework, or keep local modifications, you do not have to reissue VMFAPPLY.

   Do not erase $VMFAPP $ERRLOG.

MAP 0011: What to Do Next

001

Are you also servicing CMS?
Yes **No**

002

– Go to Chapter 14, "Rebuilding CP after Applying Service" on page 515.

003

– Go to Chapter 13, "Rebuilding CMS after Applying Service" on page 457.

Chapter 13. Rebuilding CMS after Applying Service

This chapter describes:

- A step-by-step procedure for rebuilding CMS after applying program update service, corrective service, or local service.

MAP 0012: Should You Be Doing This Now?

001

Are you also servicing CP?
Yes **No**

002

– Continue with this chapter.

003

Have you run VMFAPPLY to apply service to CP?
Yes **No**

004

– Go to Chapter 12, "Applying Program Update Service or Corrective Service to CP" on page 453.

005

– Continue with this chapter.

Step 1. Build a New CMS Macro Library

1. Establish the appropriate minidisk access order:

```
setup ■
vmfsetup 56643082 cms (bld ■
```

Program Update Service Only

2. If you are processing program update service, erase CMSNEW MACLIB if it exists. **Do not erase CMSNEW MACLIB if you are processing corrective service or local service.**

```
erase cmsnew maclib fm ■
Ready; T=n.nn/n.nn hh:mm:ss
```

End of Program Update Service Only

	Do the rest of this step if: • **You are processing program update service and have local modifications to CMS macros or control blocks** • **You are processing corrective service** • **You are processing local service.**	

3. Access 395 as A:

```
access 395 a ■
```

4. Find out which macros and control blocks have been updated by IBM:

```
listfile * T*DS fm (exec ■
rename cms exec a cmsmcfix = = ■
xedit cmsmcfix exec ■
sort * 1 50 ■
file ■
```

The updates you have just received are on the alternate LOCAL1 (395) disk if you are applying corrective service, the alternate DELTA (593) disk if you are applying program update service. You may also have updates on other disks in the LOCAL1 string.

The **exec** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

5. Find out which disk the source files for macros and control blocks are on. They should be on the CMS BASE2 disk (393).

```
listfile * macro * ■
Ready; T=n.nn/n.nn hh:mm:ss
listfile * copy * ■
Ready; T=n.nn/n.nn hh:mm:ss
```

6. Create or edit an EXEC to generate a macro library for updated macros called CMSNEW EXEC. It should have the same format as DMSSP EXEC, and should list:
   - For program update service, only those macros and control blocks for which you have received service and for which you also have local modifications
   - For corrective service, all macros and control blocks for which you have received service, plus any for which you have local modifications
   - For local service, all macros and control blocks for which you have local modifications.

   If you have no macros or control block files that must be listed in CMSNEW EXEC, go to substep 11 on page 460.

   a. If you have not already created CMSNEW EXEC, issue:

   ```
   copy dmssp exec fm cmsnew = a (recfm f lrecl 80 ■
   ```

   b. If you already have a copy of CMSNEW EXEC, issue:

   ```
   copy cmsnew exec fm = = a (recfm f lrecl 80 ■
   ```

   Now edit CMSNEW EXEC, adding all necessary macro and control block names. If you created CMSNEW EXEC by copying DMSSP EXEC, use the contents of DMSSP EXEC only for a guide to the format of the macro entries. After you have added the macro and control block names you need, in the proper format, delete the macro names that came from DMSSP EXEC.

   ```
   xedit cmsnew exec a ■
   :
   file ■
   ```

7. You now need to copy every macro and control block file listed in CMSNEW EXEC to your A-disk.

   ```
   copy fn macro fm = = a (olddate replace ■
   copy fn copy fm = = a (olddate replace ■
   ```

8. Generate the new macro library by using the VMFMAC EXEC:

   ```
   vmfmac cmsnew dmsmxa ■
   DMSUPD178I Updating macroname MACRO A1
   DMSUPD178I Applying macroname update A1
   macroname MACRO ADDED
   :
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   DMSMXA CNTRL is a special control file used for generating CMS macro libraries. Do not confuse it with DMSXA CNTRL, the CMS main control file.

   Note: An alternate method to using the VMFMAC EXEC is to update the control blocks and macros by using the UPDATE module, issuing MACLIB DELETE and then MACLIB ADD commands for the MACLIB containing the changed macro or control block.

   See *VM/XA SP CMS Command Reference* for details on the MACLIB command.

9. The new macro library is now on your A-disk (395). You can now erase CMSNEW COPY (a file created by VMFMAC) from the 395 disk.

   ```
   erase cmsnew copy a ■
   ```

10. Add the new macro library name to the TEXT MACS card in the DMSXA CNTRL file. (DMSXA CNTRL is the the CMS main control file. Do not confuse it with DMSMXA CNTRL, used for generating CMS macro libraries.)

    Make the new macro library the first one listed so that you use the updated macros.

    ```
    xedit dmsxa cntrl ■
    locate/TEXT MACS ■
    TEXT MACS maclib ...
    change/MACS/MACS CMSNEW/ ■
    TEXT MACS CMSNEW maclib ...
    file ■
    ```

11. Access 395 as B and 191 as A:

    ```
    access 395 b ■
    access 191 a ■
    ```

Step 2. Build a New CP Macro Library

1. Establish the appropriate minidisk access order:

```
setup ■
Ready; T=n.nn/n.nn hh:mm:ss
vmfsetup 56643082 cp (bld ■
DMSWSU1900W The existing 56643082 $SETUP A1 file
             has been refreshed.  You might want to
             check your access order when done.
Ready(0004); T=n.nn/n.nn hh:mm:ss
```

Program Update Service Only

2. If you are processing program update service, erase CPNEW MACLIB if it exists. **Do not erase CPNEW MACLIB if you are processing corrective service or local service.**

```
erase cpnew maclib fm ■
Ready; T=n.nn/n.nn hh:mm:ss
```

End of Program Update Service Only

	Do the rest of this step if: • **You are also servicing CP, *and*** • **One of these conditions applies:** – **You are processing program update service and have local modifications to CP macros or control blocks** – **You are processing corrective service** – **You are processing local service.**	

3. Access 295 as A:

```
access 295 a ■
```

4. Find out which macros and control blocks have been updated by IBM. First list all the CP updates you received from IBM:

```
listfile * H*HP fm (exec ■
rename cms exec a cpupdate = = ■
xedit cpupdate exec ■
sort * 1 50 ■
file ■
```

The updates are on the CP alternate LOCAL1 (295) disk if you are applying corrective service, or on the CP alternate DELTA (594) disk if you are applying program update service. You may also have updates on other disks in the LOCAL1 string.

The **exec** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

5. Find which of the updates you received affect macros and control blocks. At the same time, note where those macros and control blocks are. The source files for macros and control blocks should be on the CP BASE2 disk (394). For each update listed in CPUPDATE EXEC, issue:

```
listfile fn macro * ■
Ready; T=n.nn/n.nn hh:mm:ss
listfile fn copy * ■
Ready; T=n.nn/n.nn hh:mm:ss
```

If the update affects a macro or control block file, that file will be listed.

6. Create or edit an EXEC called CPNEW EXEC to generate a library for updated macros. It should have the same format as HCPXA1 EXEC, and should list:
   - For program update service, only those macros and control blocks for which you have received service and for which you also have local modifications
   - For corrective service, all macros and control blocks for which you have received service, plus any for which you have local modifications
   - For local service, all macros and control blocks for which you have local modifications.

   If you have no macros or control block files that must be listed in CPNEW EXEC, go to substep 11 on page 463.

   a. If you have not already created CPNEW EXEC, issue:

```
copy hcpxa1 exec fm cpnew = a (recfm f lrecl 80 ■
```

   b. If you already have a copy of CPNEW EXEC, issue:

```
copy cpnew exec fm = = a (recfm f lrecl 80 ■
```

   Now edit CPNEW EXEC, adding all necessary macro and control block names. If you created CPNEW EXEC by copying HCPXA1 EXEC, use the contents of HCPXA1 EXEC only for a guide to the format of the macro entries. After you have added the macros and control block names you need, in the proper format, delete the macro names that came from HCPXA1 EXEC.

```
xedit cpnew exec a ■
⋮
file ■
```

7. You now need to copy every macro and control block file listed in CPNEW EXEC to your A-disk.

```
copy fn macro fm = = a (olddate replace ■
copy fn copy fm = = a (olddate replace ■
```

8. Generate the new macro library by using the VMFMAC EXEC:

```
vmfmac cpnew hcpxa ■
DMSUPD178I Updating macroname MACRO A1
DMSUPD178I Applying macroname update A1
macroname MACRO ADDED
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

Note: An alternate method to using the VMFMAC EXEC is to update the control blocks and macros by using the UPDATE module, and issuing MACLIB DELETE and then MACLIB ADD commands for the MACLIB containing the changed macro or control block.

See *VM/XA SP CMS Command Reference* for details on the MACLIB command.

9. The new macro library is now on your A-disk (295). You can now erase CPNEW COPY (a file created by VMFMAC) from the 295 disk.

```
erase cpnew copy a ■
```

10. Add the new macro library name to the TEXT MACS card in the HCPXA CNTRL file. Make it the first macro library listed so that you use the updated macros.

```
xedit hcpxa cntrl ■
locate/TEXT MACS ■
TEXT MACS HCPXA1 ...
change/MACS/MACS CPNEW/ ■
TEXT MACS CPNEW HCPXA1 ...
file ■
```

11. Reaccess 295 as B and 191 as A:

```
access 295 b ■
access 191 a ■
```

Step 3. Update the CMS Message Repository

	Do this step if:	
	• **You are processing program update service and have local modifications to the CMS message repository** • **You are processing corrective service and have local modifications to the CMS message repository** • **You are processing local service.**	

1. Determine your system default national language:

   ```
   query lang ■
   ```

   langid — See Table 5 on page 317 to identify the language corresponding to *langid*. Note the country code for that language.

2. Invoke the VMFNLS EXEC to update the CMS message repository:

   ```
   vmfnls dmsmesy repos 56643082 cms ■
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   y is the country code for your system national language.

3. If you have updates to the message repositories for any other national languages installed on your system, repeat substep 2 for each language.

Step 4. Assemble the Changed ASSEMBLE Files

	Do this step if: • **You are processing program update service and have local modifications to ASSEMBLE files** • **You are processing corrective service** • **You are processing local service.**	

1. Establish the appropriate minidisk access order:

   setup ■

2. Review the apply exception log ($VMFAPP $ERRLOG) for the names of ASSEMBLE files to which you have applied service and for which you also have local updates. Chapter 11, "Applying Program Update Service or Corrective Service to CMS" on page 449 has instructions for deciding which of these local updates you wish to keep.

   vmfview apply ■

   To find these ASSEMBLE files, search for all instances of message DMSxxx1885W. This message lists all the MACRO, COPY, EXEC, XEDIT, and ASSEMBLE files for which you have local updates. You must reassemble any such ASSEMBLE files to pick up the local updates.

	If you have applied IBM service to any ASSEMBLE files for which you have local updates, and you wish to keep the local updates, continue with substep 3. Otherwise, go to "Step 5. Build the Nucleus" on page 467.	

3. Copy the ASSEMBLE files you need to reassemble from the 393 disk to your A-disk:

   copy dms*xxx* assemble 1 = = a (unpack olddate ■
   Ready; T=*n.nn/n.nn hh:mm:ss*

   Substitute the last 3 letters of the filename of the ASSEMBLE file for *xxx*.

4. Use the VMFHASM EXEC to update and assemble all the files for which you have both IBM service and local modifications:

   vmfhasm dms*xxx* 56643082 cms ■

   Substitute the last 3 letters of the filename of the ASSEMBLE file for *xxx*.

   DMSUPD178I UPDATING DMS*xxx* ASSEMBLE *filemode*
   DMSUPD178I APPLYING DMS*xxx* *filetype filemode*
   ASSEMBLING DMS*xxx*
   DMS*xxx* {TEXT|*xxxnnnnn*} CREATED
   PRT FILE *nnnn* SENT FROM MAINT PRT AS *nnnn*
   RECS *nnnn* COPY 001 I NOHOLD NOKEEP
   Ready; T=*n.nn/n.nn hh:mm:ss*

5. Copy all the text files from the A-disk to the alternate LOCAL1 (395) disk:

   copy dms*xxx* {**text**|*xxxnnnnn*} **a = = b** ■

6. If you reassemble any files that are not in the CMS load list, copy the text decks to the alternate LOCAL1 disk again, giving the copy the filetype TEXT:

   copy dms*xxx* {**text**|*xxxnnnnn*} **a = text b** ■

7. Erase all the ASSEMBLE files and text files from your A-disk:

   erase dms*xxx* **assemble a** ■
   erase dms*xxx* {**text**|*xxxnnnnn*} **a** ■

8. Make sure that you perform substeps 3 on page 465 through 7 for each file that must be reassembled.

Step 5. Build the Nucleus

1. Establish the appropriate minidisk access order:

   ```
   setup ■
   ```

2. Generate the new CMS nucleus:

   ```
   spool punch * class n ■
   spool prt * ■
   spool rdr class n ■
   ```

   ```
   vmfbld 56643082 cms (punch ■
   ```

   If you are processing PUT service, use **cms** as the component name.

   or

   ```
   vmfbld 56643082 corcms (punch ■
   ```

   If you are processing COR service, use **corcms** as the component name.

   ```
   DMSBLD1851I Processing CMSLOAD with
               the part handler VMFBDNUC EXEC
   LOAD LIST: $$$TLL$$ EXEC A1 (MNT191)
   ```

   (To ensure that text decks are copied to the correct disk during VMFBLD execution using VMFBDCPY, the component name for CMS corrective service is now corcms. This causes the corrective service override to be applied to the 56643082 $PPF and the DELTA disk string to be redefined).

   This invokes the VMFBLD EXEC to build a new nucleus for CMS. The VMFBLD EXEC performs a number of system generation functions for you. For more information on this EXEC, see "VMFBLD EXEC" on page 679.

   ```
   RDR FILE fileno SENT FROM MAINT PUN AS fileno
      RECS nnnn COPY 001 N NOHOLD NOKEEP
   ```

   The system loader punches the CMS nucleus. This message tells you that the punch file is complete.

   ```
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   If you get return code 4, the reason may be that a PTF which is now being applied has another PTF, outside CMS, as a prerequisite. Check the $VMFBLD $ERRLOG file for message DMSBNC1854W.

3. Review the build exception log ($VMFBLD $ERRLOG). If necessary, correct any problems before going on.

   ```
   vmfview build ■
   ```

4. IPL your virtual reader:

   ```
   ipl 00c ■
   ```

```
DMSINS327I  The installation saved segment could not be loaded.
```

Informational message. The CMSINST installation segment has not been loaded and saved yet.

```
VM/XA ALTCMS mm/dd/yy
■
SYNONYM SYN
CP TERM MODE VM
```

The version identification you defined appears here and each time that you IPL 490 or IPL CMS.

```
Ready; T=n.nn/n.nn hh:mm:ss
```

The ready message indicates that the nucleus has been loaded to your 490 minidisk successfully.

5. Query the reader to identify the CMS load map:

```
query rdr * all ■
```

```
ORIGINID FILE   CLASS RECORDS  CPY HOLD DATE  TIME     NAME     TYPE DIST
MAINT    fileno M PUN nnnnnnnn 001 NONE mm/dd hh:mm:ss              SYSPROG
```

The CMS load map is the file with a blank filename and filetype. It has approximately 8100 records. The exact size varies according to the system default language, local modifications, and VMSUP level. Note the *fileno* of this file. You will use it in your next command.

6. Receive the CMS load map that is in your reader.

```
spool prt * nohold ■
close prt * ■
```

```
setup ■
receive fileno cmsnuc map b (replace ■
CMSNUC MAP B1 created
DMSRDC738I Record length is 132 bytes
File CMSNUC MAP B received from MAINT at
  nodeid as CMSNUC MAP B
Ready; T=n.nn/n.nn hh:mm:ss
```

The CMS load map is loaded onto the CMS LOCAL1 (0395) minidisk.

7. Review the CMS load map to verify that any local modifications to CMS modules have been applied.

	Warning: From this point on, use IPL 490 instead of IPL 190 whenever you need to re-IPL CMS. The alternate (test) nucleus is on the 490 disk; the current (production) nucleus is on the 190 disk.	

Step 6. Generate Executable Modules

	Do this step if:	
	• **You are processing program update service and have local modifications to executable modules** • **You are processing corrective service** • **You are processing local service.**	

1. Establish the appropriate minidisk access order:

```
setup ■
vmfsetup 56643082 cms (bld ■
```

2. Determine if you need to generate any executable modules.

   - If you are processing corrective service, make and sort a list of the files you are servicing:

```
listfile dms* text b (exec label ■
rename cms exec a dmstext = = ■
xedit dmstext exec ■
sort * 1 50 ■
file ■
```

     The **exec** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

     Check the whole list against Table 12. You must generate an executable module for any file you find listed there.

   - If you are processing program update service, review the apply exception log for files that have both IBM service and local modifications.

```
vmfview apply ■
```

     To find these files, search the exception log for those messages that have message prefix BD: and message number DMSxxx1885W. Check these files against Table 12. You must generate an executable module for any file you find listed there.

3. Find each executable module you have to regenerate in the first column of the table. Type the command in the second column. Make sure that the file named in the third column is created. You can ignore any "INVALID CARD" messages you receive while running CMSGEND.

   During regeneration of the modules in Table 12, files with a filetype of LKEDIT may be created and erased. You do not need them.

Table 12 (Page 1 of 5). CMSGEND Files

Assemble File Name	Command to Type In	File Created by this Command
Note: If you receive service for a CMS text deck not listed in this table, it will be regenerated in some other way. You do not need to regenerate it now. (See Table 15 on page 589 in Appendix A, "VM/XA System Product Regeneration Requirements" for the procedure used. Text decks not listed in this table or in Table 15 on page 589 become part of the nucleus when the system is built if they are listed in the CMS load list.)		
DMSAMS	**cmsgend amserv ■**	AMSERV MODULE

Table 12 (Page 2 of 5). CMSGEND Files

Assemble File Name	Command to Type In	File Created by this Command
DMSASD, DMSASM	**cmsgend assemble** ■	ASSEMBLE MODULE
DMSASN	**cmsgend assgn** ■	ASSGN MODULE
DMSBCT, DMSBLG, DMSCDI, DMSDFT, DMSSMG, DMSUSR	**cmsgend dmsdft** ■	DMSDFT MODULE
DMSBOF, DMSBUS, DMSCCM, DMSCIA, DMSICT, DMSPBS, DMSSAP, DMSSUP, DMSTRC	**cmsgend dmscut** ■	DMSCUT MODULE
DMSBTB	**cmsgend cmsbatch** ■	CMSBATCH MODULE
DMSCCK	**cmsgend catcheck** ■	CATCHECK MODULE
DMSCMP	**cmsgend compare** ■	COMPARE MODULE
DMSDLK	**cmsgend doslked** ■	DOSLKED MODULE
DMSDSK	**cmsgend disk** ■	DISK MODULE
DMSDSL	**cmsgend doslib** ■	DOSLIB MODULE
DMSDSV	**cmsgend dserv** ■	DSERV MODULE
DMSEDC, DMSEDF, DMSEDI, DMSEDX, DMSGIO, DMSSCR, DMSZIT	**cmsgend edit** ■	EDIT MODULE (see Note 1 on page 473)
DMSEXG	**cmsgend dcssgen** ■	DCSSGEN MODULE
DMSEXM	**cmsgend execmap** ■	EXECMAP MODULE
DMSFOR	**cmsgend format** ■	FORMAT MODULE
DMSGLB	**cmsgend global** ■	GLOBAL MODULE
DMSGND	**cmsgend gendirt** ■	GENDIRT MODULE
DMSHLB, DMSHLD, DMSHLI, DMSHLP, DMSHLS	**cmsgend helpconv** ■	HELPCONV MODULE

Table 12 (Page 3 of 5). CMSGEND Files

Assemble File Name	Command to Type In	File Created by this Command
DMSICP, IOPC*xxxx*, IOPP*xxxx*	**cmsgend iocp** ■	IOCP MODULE
DMSIMA	**cmsgend imagemod** ■	IMAGEMOD MODULE
DMSLBD	**cmsgend labeldef** ■	LABELDEF MODULE
DMSLBM	**cmsgend maclib** ■	MACLIB MODULE
DMSLBT	**cmsgend txtlib** ■	TXTLIB MODULE
DMSLDS	**cmsgend listds** ■	LISTDS MODULE
DMSLLU	**cmsgend listio** ■	LISTIO MODULE
DMSLMX	**cmsgend tape** ■	TAPE MODULE (see Note 2 on page 473)
DMSMDP	**cmsgend modmap** ■	MODMAP MODULE
DMSMGC, DMSMGD, DMSMGE	**cmsgend genmsg** ■	GENMSG MODULE
DMSMVE	**cmsgend movefile** ■	MOVEFILE MODULE
DMSNXD	**cmsgend nucxdrop** ■	NUCXDROP MODULE
DMSOPT	**cmsgend option** ■	OPTION MODULE
DMSOSR	**cmsgend osrun** ■	OSRUN MODULE
DMSOVR	**cmsgend svctrace** ■	SVCTRACE MODULE
DMSOVS	**cmsgend dmsovs** ■	DMSOVS MODULE
DMSPCA, DMSPCB, DMSPCC, DMSPCR, DMSPCT, DMSPCW	**cmsgend dmspcc** ■	DMSPCC MODULE
DMSPOA, DMSPOC, DMSPOD, DMSPOE, DMSPOL, DMSPON, DMSPOP, DMSPOQ, DMSPOR, DMSPOS	**cmsgend prop** ■	PROPLIB LOADLIB (see Note 3 on page 473)
DMSPRE	**cmsgend preload** ■	PRELOAD MODULE
DMSPRV	**cmsgend pserv** ■	PSERV MODULE
DMSPUN	**cmsgend punch** ■	PUNCH MODULE
DMSRDC	**cmsgend readcard** ■	READCARD MODULE

Table 12 (Page 4 of 5). CMSGEND Files

Assemble File Name	Command to Type In	File Created by this Command
DMSRDR	**cmsgend rdr** ■	RDR MODULE
DMSRNE	**cmsgend renum** ■	RENUM MODULE
DMSRRV	**cmsgend rserv** ■	RSERV MODULE
DMSRSV	**cmsgend reserve** ■	RESERVE MODULE
DMSSFD	**cmsgend savefd** ■	SAVEFD MODULE
DMSSNC	**cmsgend dmssnc** ■	DMSSNC MODULE
DMSSPR	**cmsgend setprt** ■	SETPRT MODULE
DMSSRT	**cmsgend sort** ■	SORT MODULE
DMSSRV	**cmsgend sserv** ■	SSERV MODULE
DMSSSK	**cmsgend setkey** ■	SETKEY MODULE
DMSSYN	**cmsgend synonym** ■	SYNONYM MODULE
DMSTMA	**cmsgend tapemac** ■	TAPEMAC MODULE
DMSTPD	**cmsgend tappds** ■	TAPPDS MODULE
DMSTPE DMSTPF, DMSTPG, DMSTPH, DMSTPI, DMSTPJ,	**cmsgend tape** ■	TAPE MODULE (see Note 2 on page 473)
DMSTYP	**cmsgend type** ■	TYPE MODULE
DMSUPD	**cmsgend update** ■	UPDATE MODULE
DMSUTL	**cmsgend loadlib** ■	LOADLIB MODULE
DMSXMS	**cmsgend dmsxms** ■	DMSXMS MODULE (see Note 4 on page 473)
DMSXRE	**cmsgend dmsxre** ■	DMSXRE MODULE (see Note 4 on page 473)
DMSZAP	**cmsgend zap** ■	ZAP MODULE
VMFCLEAR	**load vmfclear (origin trans nomap type** ■ **genmod vmfclear module a (nomap system nostr all** ■	VMFCLEAR MODULE
VMFDATE	**cmsgend vmfdate** ■	VMFDATE MODULE
VMFDOS	**cmsgend vmfdos** ■	VMFDOS MODULE

Table 12 (Page 5 of 5). CMSGEND Files

Assemble File Name	Command to Type In	File Created by this Command
VMFLOAD	**cmsgend vmfload** ■	VMFLOAD MODULE

Notes:

1. When the CMSGEND EXEC procedure is invoked for EDIT, it creates the EDIT module. Then it automatically reinvokes itself to create the EDMAIN module.
2. When the CMSGEND EXEC is invoked for TAPE, it creates the TAPE module and then reinvokes itself to create the DMSLMX and DMSTP*x* modules.
3. You get messages DMSSLK0008W and DMSSOP036E when you regenerate PROPLIB. You can ignore them.
4. All EDIT source files, except DMSXMS and DMSXRE, are contained within the CMS nucleus.

4. Test any changed CMS commands.

Step 7. Regenerate System Product Interpreter Programs

	Do this step if:	
	• **You are processing program update service and have local modifications to System Product Interpreter programs (EXECs and XEDIT macros)** • **You are processing corrective service** • **You are processing local service.**	

1. Establish the appropriate minidisk access order:

   ```
   setup ■
   vmfsetup 56643082 cms (bld ■
   ```

2. Find out whether you have received service for any System Product Interpreter programs (EXECs or XEDIT macros):

   ```
   listfile * EXEC fm (exec ■
   rename cms exec a execfix = = ■
   listfile * XEDIT fm (exec ■
   rename cms exec a xeditfix = = ■
   ```

   The updates are on the alternate LOCAL1 (395) disk if you are applying corrective service, or on the alternate DELTA (593) disk if you are applying program update service.

   The **exec** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

3. Examine the list of EXECs and XEDIT macros to which you have applied service and find out whether you have local modifications (including corrective service not included on the PUT you are processing) for any of them.

   Warning: The $VMFAPP $ERRLOG file will not identify these EXECs and XEDIT macros for you. You must know what local modifications you have made.

4. Copy the latest version of the base code for each EXEC or XEDIT macro for which you have both IBM service and local modifications to the CMS alternate LOCAL1 disk (395).

   ```
   copy fn {$EXEC|$XEDIT} fm = = b ■
   ```

5. Regenerate any EXECs or XEDIT macros for which you have received service and for which you have local modifications:

   ```
   execupdt fn ft b (ctl dmsxa options ■
   ```

   ft is EXEC or XEDIT, *without the dollar sign.* DMSXA is the CMS control filename.

   For any files with a SID code in columns 63 through 71, use the HIST and SID options. For all other files, including the following:

   PROPEPIF EXEC
   PROPHCHK EXEC
   PROPLGER EXEC
   PROPPCHK EXEC
   PROPPROF EXEC
   PROPRTCV EXEC
   PROPST EXEC

   use only the HIST option.

6. Erase the base code from the CMS alternate LOCAL1 disk:

   erase *fn* {**$EXEC**|**$XEDIT**} **b** ■

7. Repeat substeps 4 on page 474 through 6 for each System Product Interpreter program that needs regenerating.

Step 8. Copy Changed Files to 0490

1. Issue IPL 490 CLEAR.

   ipl 490 clear ■
   VM/XA ALTCMS *mm/dd/yy*
   ■
   Ready; T=*n.nn/n.nn hh:mm:ss*

2. Establish the appropriate minidisk access order:

   setup ■
   vmfsetup 56643082 cms (all ■

	Do the next substep if you want to make the macro libraries you created in Step 1 and Step 2 available to general users.	

3. If you want to make either CMSNEW MACLIB or CPNEW MACLIB available to general users, copy it to the alternate CMS BUILD1 disk (the M-disk).

   copy *fn* **maclib** *fm* **= = m2 (olddate replace** ■

	For each new executable module created by running the CMSGEND EXEC in Step 6, repeat the next two substeps.	

4. Save the new executable modules on MAINT's 490 disk:

   copy *fn ft* **a = = m2 (olddate replace** ■
   Ready; T=*n.nn/n.nn hh:mm:ss*

   Substitute for *fn* and *ft* the name listed in the rightmost column of Table 12 on page 469 that corresponds to the ASSEMBLE file that you serviced.

   The replace option on the COPY command causes the command to overlay the file (*fn* MODULE) on the 490 minidisk, if it already exists there.

5. Erase the copy that is still out on the 191 disk:

   erase *fn ft* **a** ■
   Ready; T=*n.nn/n.nn hh:mm:ss*

	For each EXEC or XEDIT macro you serviced, whether you regenerated it in Step 7 or not, repeat the next two substeps.	

6. Check to see if the original EXEC or XEDIT macro was on MAINT's 490 disk:

   listfile *fn ft* **m** ■

7. If the replaced or updated EXEC or XEDIT macro belongs on MAINT's 490 disk, copy it there:

```
copy fn ft fm = = m2 (olddate replace ■
Ready; T=n.nn/n.nn hh:mm:ss
```

The macros are on the alternate LOCAL1 (395) disk if you are applying corrective service, the alternate DELTA (593) disk if you are applying program update service.

8. Make backup copies of all your alternate service minidisks and of your current DELTA, APPLY, and LOCAL minidisks, using the DASD Dump Restore program.

Step 9. Create Test EXECs and Files

In this step, you will create the EXECs and files you need to create test named saved systems and saved segments.

1. Choose temporary names for all your new named saved systems and saved segments so that you do not overlay those you are working with until you are satisfied that the new ones work. This sample procedure uses the following names:

Real name	Temporary name
CMS	ALTCMS
CMSXA	ALTCMSXA
DOSBAM	ALTDSBAM
DOSINST	ALTDOSIN
CMSDOS	ALTDOS
CMSBAM	ALTBAM
CMSVSAM	ALTVSAM
CMSAMS	ALTAMS
INSTHELP	ALTINSHP
CMSINST	ALTINST
HELP	ALTHELP
GCS	ALTGCS

2. Copy the SAMPNSS EXEC:

```
setup ■
copy sampnss exec fm testnss = = ■
```

3. Edit the TESTNSS EXEC, changing all the real names to temporary names.

```
xedit testnss exec ■
:
```

When you have finished, the TESTNSS EXEC should look like this:

```
/*************************************************************/
/**     Virtual Machine / System Product       5664-308    **/
/**     Contains restricted materials of IBM               **/
/**     Copyright (c) I B M   Corporation          1988    **/
/**     Licensed Materials -  Property of I B M            **/
/**     Refer to Copyright Instructions: Form G120-2083    **/
/*************************************************************/
arg parm1 parm2
Address Command
Select
  when parm1 = 'ALTCMS' & parm2 = 'ALTCMSXA' then
    do;
      'CP DEFSYS ALTCMS   0-A EW 20-22  EW E00-FFF SR MINSIZE=256K'
      'CP DEFSYS ALTCMSXA 0-A EW 20-22  EW E00-FFF SR MINSIZE=256K'
    end;
  WHEN PARM1 = 'ALTGCS' THEN
    'CP DEFSYS ALTGCS  0-6 EW 400-5FF SW MINSIZE=256K VMGROUP RSTD'
  when parm1 = 'ALTINST' then
    'CP DEFSEG ALTINST C00-C4F SR SPACE ALTINSHP'
  WHEN PARM1 = 'ALTHELP' THEN
    'CP DEFSEG ALTHELP C50-C9F SR SPACE ALTINSHP'
```

```
  WHEN PARM1 = 'ALTDOS' THEN
    'CP DEFSEG ALTDOS B00-B0F SR SPACE ALTDSBAM'
  WHEN PARM1 = 'ALTBAM' THEN
    'CP DEFSEG ALTBAM B10-B3F SR SPACE ALTDSBAM'
  WHEN PARM1 = 'ALTVSAM' THEN
    'CP DEFSEG ALTVSAM BA0-BFF SR A30-A3F EW SPACE ALTDSBAM'
  WHEN PARM1 = 'ALTAMS' THEN
    'CP DEFSEG ALTAMS B40-B9F SR A00-A2F EW SPACE ALTDSBAM'
  Otherwise
    do;
      say 'Unrecognized Parameters passed - ' parm1 parm2
    end;
End  /* Select */
'CP QUERY NSS ALL'
Exit
```

4. File the TESTNSS EXEC:

   file ■

5. Copy the latest version of the SAMGEN EXEC:

   copy samgen exec *fm* **altsamgn = =** ■

6. Edit the ALTSAMGN EXEC, changing CMSBAM to ALTBAM throughout.

   xedit altsamgn exec ■
   change/CMSBAM/ALTBAM/* * ■
   file ■

7. Copy CMSBAM MAP and CMSBAM DOSLIB as ALTBAM MAP and ALTBAM DOSLIB:

   copy cmsbam map *fm* **altbam = =** ■
   copy cmsbam doslib *fm* **altbam = =** ■

8. Copy the latest version of the VSAMGEN EXEC:

   copy vsamgen exec *fm* **altvsamg = =** ■

9. Edit the ALTVSAMG EXEC, changing CMSVSAM to ALTVSAM and CMSAMS to ALTAMS throughout.

   xedit altvsamg exec ■
   change/CMSVSAM/ALTVSAM/* * ■
   top ■
   change/CMSAMS/ALTAMS/* * ■
   file ■

10. Copy all files with a filename of CMSVSAM, CMSAMS, or CMSAMS*x* giving the copies filetypes of ALTVSAM, ALTAMS, and ALTAMS*x*:

    filelist cmsvsam * * ■
    copy cmsvsam *ft fm* **altvsam = =** ■
    filelist cmsams* * * ■
    copy cmsams *ft fm* **altams = =** ■
    copy cmsams*x* *ft fm* **altams***x* **= =** ■

11. You may have created a load list for the DCSSGEN command during the installation process. The one created in the sample installation process was called INSTLIST FILE. It should be on your A-disk (191). If it is not there, you probably used the IBM-supplied load list. This load list is called CMSINST EXECLIST. It should be on your 193 disk. Copy one of these load lists as ALTINST EXECLIST:

```
copy fn ft fm altinst execlist k ■
```

Step 10. Build Test Named Saved Systems

1. Use the TESTNSS EXEC to define segments and save your new CMS and CMSXA:

```
testnss altcms altcmsxa ■
```

The TESTNSS EXEC issues the DEFSYS command.

```
HCPNSD440I The Named Saved System (NSS) ALTCMS
            was successfully defined in fileid
            fileno.
HCPNSD440I The Named Saved System (NSS) ALTCMSXA
            was successfully defined in fileid
            fileno.
```

Note that the names are positional on the TESTNSS command (the System/370 name followed by the 370-XA name).

The TESTNSS EXEC first issues the DEFSYS command to define a skeleton system data file for ALTCMS and for ALTCMSXA. These messages tell you the DEFSYS commands were successful.

```
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
⋮
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss ALTCMS   NSS      MAINT
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss ALTCMSXA NSS      MAINT
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

The TESTNSS EXEC issues a QUERY NSS command. These messages show what system data files are defined.

2. Issue the QUERY NSS ALL MAP command to make sure ALTCMS is defined properly. Check to see whether the information under BEGPAG, ENDPAG, TYPE, and CL in the response to QUERY NSS ALL MAP is the same as shown.

```
query nss all map ■
FILE FILENAME FILETYPE MINSIZE  BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
⋮
nnnn ALTCMS   NSS      000256K  00000  0000A   EW  S  00000  OMITTED  NO
                                00020  00022   EW
                                00E00  00FFF   SR
nnnn ALTCMSXA NSS      000256K  00000  0000A   EW  S  00000  OMITTED  NO
                                00020  00022   EW
                                00E00  00FFF   SR
⋮
```

3. Set your machine mode to 370, load CMS (IPL 490), and save the ALTCMS system.

```
set machine 370 ■
```

```
ipl 490 clear parm savesys altcms ■
```

Load 490 with the option to save the system under the name ALTCMS.

```
HCPNSS440I The named saved system ALTCMS was
            successfully saved in fileid fileno.
VM/XA ALTCMS mm/dd/yy
```

■ Press **ENTER** to initialize ALTCMS.

```
Ready; T=n.nn/n.nn hh:mm:ss
```

4. Set your machine mode to XA, load CMS (IPL 490), and save the ALTCMSXA system.

```
set machine xa ■
```

```
ipl 490 clear parm savesys altcmsxa ■
```

Load 490 with the option to save the system under the name ALTCMSXA.

```
HCPNSS440I The named saved system ALTCMSXA was
             successfully saved in fileid fileno.
VM/XA ALTCMS mm/dd/yy
```

■ Press **ENTER** to initialize ALTCMSXA.

```
Ready; T=n.nn/n.nn hh:mm:ss
```

5. Set your machine mode to 370 (or XA), redefine your virtual storage to 2 megabytes, and load the ALTCMS (or ALTCMSXA) named saved system. The example below shows loading ALTCMS in System/370 mode:

```
set machine 370 ■
define storage 2m ■
```

```
STORAGE =      2M
STORAGE CLEARED - SYSTEM RESET
ipl altcms ■
VM/XA ALTCMS mm/dd/yy
```

Defining storage causes a system reset.

■

```
Ready; T=n.nn/n.nn hh:mm:ss
```

After receiving the version identification, press **ENTER** to complete ALTCMS or ALTCMSXA initialization.

6. Check the ALTCMS status by issuing QUERY NSS ALL and QUERY NSS ALL MAP. Check to find out whether the information under FILENAME and FILETYPE in the response to QUERY NSS ALL is the same as shown. Also, check whether the information under BEGPAG, ENDPAG, TYPE, and CL in the response to QUERY NSS ALL MAP is the same as shown. The system data file should now have class A (rather than S) and have one user (MAINT).

```
query nss all ■
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
⋮
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMS   NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMSXA NSS      MAINT
⋮
Ready; T=n.nn/n.nn hh:mm:ss
query nss all map ■
FILE FILENAME FILETYPE MINSIZE  BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
⋮
nnnn ALTCMS   NSS      000256K  00000  0000A   EW  A  00001  OMITTED  NO
                                00020  00022   EW
                                00E00  00FFF   SR
nnnn ALTCMSXA NSS      000256K  00000  0000A   EW  A  00000  OMITTED  NO
                                00020  00022   EW
                                00E00  00FFF   SR
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

	From now until you do Step 17, IPL your alternate (test) CMS system (ALTCMS or ALTCMSXA) instead of your current (production) CMS.	

Step 11. Install Test CMSDOS, CMSBAM, CMSVSAM, and CMSAMS Saved Segments

	Warning: Do not skip this step even if you have not changed CMSDOS, CMSBAM, CMSVSAM, or CMSAMS. **Warning: The test CMSDOS and CMSBAM segments must be installed before you install the test CMSVSAM and CMSAMS segments.**	

1. Define your storage as 16 megabytes and IPL your new CMS system.

```
define storage 16m ■
STORAGE = 0016M
STORAGE CLEARED - SYSTEM RESET
ipl altcms ■
DMSINS327I The installation saved segment could not be loaded
```

```
VM/XA ALTCMS mm/dd/yy
```
This is a sample version header.

** ***DO NOT press ENTER!*** **

```
access (noprof ■
```
This command suppresses execution of MAINT's PROFILE EXEC.

2. Establish the appropriate minidisk access order:

```
setup ■
```

3. Issue:

```
set emsg on ■
Ready; T=n.nn/n.nn hh:mm:ss
```
You want to see any and all error messages during execution of the installation EXEC.

4. Define a segment into which the DOSGEN EXEC can load ALTDOS.

```
defseg altdosin 900-90f sr ■
```

```
HCPNSD440I  Saved segment ALTDOSIN was successfully
              defined in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
```
This command defines a 1MB segment for ALTDOSIN starting at 900000. You may place this segment anywhere below the segment spaces defined for ALTDOS, ALTBAM, ALTVSAM, and ALTAMS.

5. Invoke the DOSGEN EXEC with a load address and the name ALTDOSIN. The load address used for ALTDOSIN in this example is 900000.

```
dosgen 900000 altdosin ■

HCPNSS440I Saved segment ALTDOSIN was successfully
            saved in fileid fileno
PRT FILE fileno SENT FROM MAINT PRT AS fileno
  RECS nnnn COPY 001 A NOHOLD NOKEEP
DMSWGN715I DOSGEN COMPLETE
Ready; T=n.nn/n.nn hh:mm:ss
```

The load address and name are those recommended for the ALTDOSIN segment. Error messages for the DOSGEN EXEC are listed on page 486.

6. Re-IPL your new CMS system:

```
ipl altcms ■
DMSINS327I The installation saved segment could not be loaded
VM/XA ALTCMS mm/dd/yy
■
SYNONYM SYN
CP TERM MODE VM
Ready; T=n.nn/n.nn hh:mm:ss
```

7. Establish the appropriate minidisk access order:

```
setup ■
```

8. Define ALTDOS, ALTBAM, ALTAMS, and ALTVSAM:

```
set sysname cmsdos altdosin ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Use **cmsdos** in this command, not **altdos**.

```
testnss altdos ■
HCPNSD440I Saved segment ALTDOS was successfully
            defined in fileid fileno
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMS   NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMSXA NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss GCS      NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTINST  DCSS     MAINT
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss ALTDSBAM DCSS     MAINT
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss ALTDOS   DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss

testnss altbam ■
HCPNSD440I Saved segment ALTBAM was successfully
            defined in fileid fileno
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMS   NSS      MAINT
⋮
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss AMTBAM   DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

```
testnss altams ■
HCPNSD440I Saved segment ALTAMS was successfully
           defined in fileid fileno
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMS   NSS      MAINT
⋮
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss ALTAMS   DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

```
testnss altvsam ■
HCPNSD440I Saved segment ALTVSAM was successfully
           defined in fileid fileno
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMS   NSS      MAINT
⋮
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss ALTVSAM  DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

9. Invoke the DOSGEN EXEC with a load address and the name ALTDOS. The load address set up by TESTNSS is B00000 for the name ALTDOS.

```
dosgen b00000 altdos ■
```

The load address and name are those recommended for the ALTDOS segment. Error messages for the DOSGEN EXEC are listed on page 486.

```
HCPNSS440I Saved segment ALTDOS was successfully
           saved in fileid fileno
PRT FILE fileno SENT FROM MAINT PRT AS fileno
  RECS nnnn COPY 001 A NOHOLD NOKEEP
DMSGEN715I DOSGEN COMPLETE
Ready; T=n.nn/n.nn hh:mm:ss
```

10. To save the load map, rename it and copy it to the 193 minidisk (K).

```
copy load map a altdos segmap k (replace ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Error Messages from DOSGEN

If DOSGEN detects an error in the address that you specified:

```
DMSGEN095E INVALID ADDRESS
```

If DOSGEN cannot find a read/write accessed A-disk:

```
DMSGEN006E NO READ/WRITE A-DISK ACCESSED.
```

If DOSGEN finds unresolved external references while loading the text files:

```
DMSGEN111E DOSGEN FAILED DUE TO LOAD ERRORS.
```

If DOSGEN detects an error while assigning the storage key or saving the segment:

```
DMSGEN412S DOSGEN FAILED DUE TO SETKEY ERRORS

DMSGEN141S DOSGEN FAILED DUE TO SAVESYS ERRORS
```

11. Replace the name CMSBAM (not ALTBAM) in the CMS SYSNAME table with any name that is NOT a name previously used as a segment name.

```
set sysname cmsbam sysname ■
Ready; T=n.nn/n.nn hh:mm:ss
```

The SET SYSNAME command enters an alternate name for CMSBAM in the SYSNAME table. Choose a name that was **not** used previously as a segment name. This command does not actually change the saved segment's name, but keeps CP from finding and IPLing the segment at the wrong time.

12. Place your CMS virtual machine in a CMS/DOS environment, then invoke ALTSAMGN to load the ALTBAM segment. You must give the EXEC an address at which to load the ALTBAM segment; this address also must match the address in the skeleton ALTBAM segment that you defined beforehand.

```
set dos on ■
```

This command places your virtual machine in a CMS/DOS environment.

```
DMSSET400I SYSTEM sysname DOES NOT EXIST
DMSSET1101I 100K DOS partition defined
              at hexadecimal location 020000.
Ready; T=n.nn/n.nn hh:mm:ss
altsamgn ■
DMSSGN363R ENTER LOCATION WHERE ALTBAM
WILL BE LOADED AND SAVED:

b10000 ■
DMSSGN364I FETCHING ALTBAM...
DMSFET710I PHASE DMSVBM ENTRY POINT AT LOCATION B100C0.
DMSSGN366R ENTER NAME OF SYSTEM TO BE SAVED:
altbam ■

HCPNSS440I Saved segment ALTBAM was successfully
             saved in fileid fileno
DMSSGN365I SYSTEM ALTBAM SAVED
Ready; T=n.nn/n.nn hh:mm:ss
```

sysname is the name you chose in substep 11.

The messages indicate that the segment has been loaded and saved.

13. Access the disk that you defined for VSAM when you were installing the system (see Step 27 in Chapter 2, Step 27 in Chapter 3, or Step 29 in Chapter 4) as your A-disk.

```
access vdevno a ■
Ready; T=n.nn/n.nn hh:mm:ss
```

This puts the minidisk (*vdevno*) in read/write mode.

14. Invoke ALTVSAMG EXEC:

```
altvsamg ■
SELECT ONE OF THE FOLLOWING FUNCTIONS BY ENTERING THE NUMBER:

 1. INSTALL AMS            (READ VSAM PRODUCT TAPE, BUILD DOSLIB, CREATE SEGMENT)
 2. INSTALL VSAM           (READ VSAM PRODUCT TAPE, BUILD DOSLIB, CREATE SEGMENT)
 3. INSTALL VSAM AND AMS   (READ VSAM PRODUCT TAPE, BUILD DOSLIB, CREATE SEGMENT)
 4. BUILD  AMS             (BUILD DOSLIB, CREATE SEGMENT)
 5. BUILD  VSAM            (BUILD DOSLIB, CREATE SEGMENT)
 6. BUILD  VSAM AND AMS    (BUILD DOSLIB, CREATE SEGMENT)
 7. RESTART AMS            (CREATE SEGMENT)
 8. RESTART VSAM           (CREATE SEGMENT)
 9. RESTART VSAM AND AMS   (CREATE SEGMENT)
10. QUIT                   (EXIT ALTVSAMG EXECUTION)

      ENTER RESPONSE...
```

```
6 ■
```

Choosing option 6 tells the EXEC to create both ALTVSAM and ALTAMS segments as new segments, without reading the text files from the VSAM product tape.

If you want to install a new release of VSAM, instead of servicing the current release, you must erase all files associated with VSAM before you issue ALTVSAMG; then, when you issue ALTVSAMG, you must choose option 3.

```
DMSVGN365R ONE OR MORE OF THE TEXT FILES LISTED IN THE CMSVSAM EXEC
            ARE MISSING.  THE VSAM PP PID TAPE SHOULD BE ON TAPE DRIVE 181
            TO RESTORE THE FILES. ENTER:
            'GO' IF TAPE DRIVE IS READY TO LOAD FILES,
            'QUIT' TO STOP GENERATION PROCESS.
go ■
```

Messages

While ALTVSAMG is processing, you may receive error and information messages. These messages are self-explanatory. Messages labeled 2101I are information messages from the linkage editor and may be ignored.

Respond to the EXEC messages as they appear:

```
DMSVGN362I LINK-EDITING ALTVSAM ...
DMSVGN363I ALTVSAM DOSLIB CREATED ON DISK 'A'.
DMSVGN370R ENTER 'GO' IF SAVED SYSTEM IS TO BE CREATED,
            OTHERWISE ENTER 'QUIT'.
go ■
DMSVGN363R ENTER LOCATION WHERE ALTVSAM SHARED WILL BE
         LOADED AND SAVED:
```

```
ba0000 ■
```

```
DMSVGN363R ENTER LOCATION WHERE ALTVSAM NONSHARED WILL BE
         LOADED AND SAVED:
a30000 ■


DMSVGN364I FETCHING ALTVSAM ...
DMSFET710I PHASE DMSVVS ENTRY POINT AT LOCATION nnnnnn
DSMVGN371R ALTVSAM IS LOADED, IF ZAPS ARE TO BE APPLIED GO INTO
        'CP' MODE, APPLY THE ZAPS AND THEN REPLY 'GO'
go ■
DMSVGN366R ENTER NAME OF SYSTEM TO BE SAVED:


altvsam ■
```

This name is the default name with which the segment was defined.

```
DMSVGN365I SYSTEM ALTVSAM SAVED.

DMSVGN368R ERASE ALTVSAM DOSLIB ? ... ENTER 'YES' OR 'NO':
no ■


DMSVGN362I LINK-EDITING ALTAMS ...
DMSVGN363I ALTAMS DOSLIB CREATED ON DISK 'A'.
DMSVGN370R ENTER 'GO' IF SAVED SYSTEM IS TO BE CREATED,
            OTHERWISE ENTER 'QUIT'.
go ■
DMSVGN363R ENTER LOCATION WHERE ALTAMS SHARED WILL BE
         LOADED AND SAVED:


b40000 ■
```

This is the recommended location.

```
DMSVGN363R ENTER LOCATION WHERE ALTAMS NONSHARED WILL BE
         LOADED AND SAVED:
a00000 ■
DMSVGN364I FETCHING ALTAMS ...
DMSFET710I PHASE DMSVAS ENTRY POINT AT LOCATION nnnnnn
DMSFET710I PHASE DMSVAN ENTRY POINT AT LOCATION nnnnnn
DMSFET710I PHASE DMSVAX ENTRY POINT AT LOCATION nnnnnn
DMSVGN371R ALTAMS IS LOADED, IF ZAPS ARE TO BE APPLIED GO INTO
        'CP' MODE, APPLY THE ZAPS AND THEN REPLY 'GO'
go ■
DMSVGN366R ENTER NAME OF SYSTEM TO BE SAVED:


altams ■
```

This name is the default name used for defining the segment.

```
DMSVGN365I SYSTEM ALTAMS SAVED.

DMSVGN368R ERASE ALTAMS DOSLIB ? ... ENTER 'YES' OR 'NO':
no ■
Ready; T=n.nn/n.nn hh:mm:ss
```

15. Set DOS off:

```
set dos off ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Note: If you loaded files from the VSAM product tape in substep 14 on page 488, do not erase them. You will need them in Step 18.

Step 12. Install Test CMSINST and HELP Saved Segments

	Warning: Do not skip this step even if you have not changed CMSINST or HELP.	

1. Make sure that you have the right NAMESAVE segments in your user directory:
   - You must have the NAMESAVE ALTHELP statement in your user directory in order to save the HELP file directory information in a test saved segment.
   - If the ALTHELP segment is defined as a member of a segment space, you must have a NAMESAVE statement (for example, NAMESAVE ALTINSHP) in the directory entry for the user who invokes SAVEFD. SAVEFD specifies the name of the segment space.

   If you have to change the directory, be sure to issue DIRECTXA to bring the changed directory online.
2. Log onto MAINT (unless you are continuing from the previous step).
3. Check your virtual storage. If it is less than 16MB, issue the following:

```
define storage 16m ■
STORAGE CLEARED - SYSTEM RESET
STORAGE = 0016M
```

4. IPL your new CMS system.

```
ipl altcms ■
DMSWSP327I The installation saved segment could not be loaded
```

```
VM/XA ALTCMS mm/dd/yy
■
SYNONYM SYN
CP TERM MODE VM
Ready; T=n.nn/n.nn hh:mm:ss
```

This is a sample version header. If you defined your own version heading, your own heading will appear.

Press **ENTER** to complete the CMS initialization.

5. Establish the appropriate minidisk access order:

```
setup ■
```

6. Release the INSTHELP segment so that you can define a test segment in the same space:

```
segment release insthelp ■
```

7. Define a segment for ALTINST.

```
testnss altinst ■
HCPNSD440I Saved segment ALTINST was successfully
            defined in fileid fileno
OWNERID FILE TYPE CL RECS DATE  TIME  FILENAME FILETYPE ORIGINID
*NSS    nnnn NSS  A  nnnn mm:dd hh:mm ALTCMS   NSS      MAINT
:
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm ALTINST  DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

8. Define the ALTHELP segment:

```
testnss althelp ■
HCPNSD440I Saved segment ALTHELP was successfully
           defined in fileid fileno
OWNERID FILE TYPE CL RECS DATE  TIME  FILENAME FILETYPE ORIGINID
*NSS    nnnn NSS  A  nnnn mm:dd hh:mm ALTCMS   NSS      MAINT
⋮
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm ALTINST  DCSS     MAINT
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm ALTHELP  DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

9. Save both segments:

```
saveseg altinst ■
```

Each member saved segment defined by SAMPNSS must be saved separately.

```
HCPNSS440I Saved segment ALTINST was successfully
           saved in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
saveseg althelp ■
HCPNSS440I Saved segment ALTHELP was successfully
           saved in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
```

10. Redefine each segment to create the skeleton segments:

```
testnss altinst ■
HCPNSD440I Saved segment ALTINST was successfully
           defined in fileid fileno
OWNERID FILE TYPE CL RECS DATE  TIME  FILENAME FILETYPE ORIGINID
*NSS    nnnn NSS  A  nnnn mm:dd hh:mm ALTCMS   NSS      MAINT
⋮
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm ALTINST  DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss

testnss althelp ■
HCPNSD440I Saved segment ALTHELP was successfully
           defined in fileid fileno
OWNERID FILE TYPE CL RECS DATE  TIME  FILENAME FILETYPE ORIGINID
*NSS    nnnn NSS  A  nnnn mm:dd hh:mm ALTCMS   NSS      MAINT
⋮
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm ALTINST  DCSS     MAINT
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm ALTHELP  DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

11. Using the load list you copied in Step 9, issue the DCSSGEN command as follows:

```
dcssgen altinst execlist k altinst ■
```

ALTINST EXECLIST K is the file ID of the file that contains the list of EXECs and editor macros to be loaded into the test segment.

ALTINST is the name of the test segment.

```
HCPNSS440I Saved segment ALTINST was successfully
           saved in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
```

Note: When you built your ALTCMS nucleus, if you indicated in the DMSNGP file (USEINST=YES) or in answer to the DMSINQ296R prompt that you wanted to use the installation segment (the default is YES), then this segment is used each time a user IPLs ALTCMS. If you previously indicated that you did not want to use the installation segment but now want to use it, you must modify the DMSNGP file to indicate that the segment should be used, then assemble the modified DMSNGP and rebuild ALTCMS.

Messages from DCSSGEN Command

While DCSSGEN is processing, you may receive error or warning messages that indicate specific conditions. If errors were encountered, after processing is complete you receive the following prompt:

```
DMSEXG298R An error has been detected while building the
           DCSS.  Do you still want the DCSS saved?
           Enter 1 (YES) or 0 (NO).
```

Enter **1** to disregard the error(s) and save the segment, or enter **0** to not save the segment. If you do not save the segment, you receive the message:

DMSEXG288I *segname* not saved

If DCSSGEN encounters an error while saving the segment, you receive the message:

DMSEXG288E *dcssname* not saved

If your virtual machine is not large enough to contain the segment (you need 16M), you receive the message:

```
DMSEXG284E The DCSS is not completely inside the virtual machine
```

To correct this situation, increase the size of your virtual machine, re-IPL ALTCMS, and reissue the DCSSGEN command.

12. Define your virtual storage **less** than the address at which the ALTHELP segment is to be loaded. For example, if the ALTHELP segment is defined at X'C50000', define your storage as 12MB.

```
define storage 12m ■
STORAGE CLEARED - SYSTEM RESET
STORAGE = 12M
```

13. Re-IPL your test CMS system.

```
ipl altcms ■
```

```
VM/XA ALTCMS mm/dd/yy                    This is a sample version header.
■
SYNONYM SYN
CP TERM MODE VM
Ready; T=n.nn/n.nn hh:mm:ss
```

14. Establish the appropriate minidisk access order:

```
setup ■
vmfsetup 56643082 cms (bld ■
```

15. Issue the following commands to initialize and save the segment:

```
savefd init vaddr label althelp ■
Ready; T=n.nn/n.nn hh:mm:ss
```

vaddr is 49D for mixed-case American English, 49C for uppercase American English.

label is the CMS label assigned to the disk. (In this sample procedure, it is MNT49D or MNT49C.)

```
savefd save vaddr label althelp ■
DMSACP723I Z(19D) R/O
HCPNSS440I Saved segment ALTHELP was successfully
           saved in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
```

16. For more information about using the SAVEFD command to save minidisk file directory information in a saved segment, refer to *VM/XA SP CMS Command Reference*. For more information about saved segments, refer to *VM/XA SP Guide to Saved Segments*.

17. Verify that the ALTINST and ALTHELP segments were defined correctly:

```
query nss all ■
OWNERID FILE   TYPE CL RECS DATE  TIME     FILENAME    FILETYPE ORIGINID
⋮
*NSS    fileno NSS  A  nnnn mm/dd hh:mm:ss ALTINSHP    NSS      MAINT
*NSS    fileno NSS  A  nnnn mm/dd hh:mm:ss ALTINST     NSS      MAINT
*NSS    fileno NSS  A  nnnn mm/dd hh:mm:ss ALTHELP     NSS      MAINT
```

18. Redefine your storage before you continue:

```
define storage 16m ■
STORAGE = 16M
Storage cleared - system reset
ipl altcms ■
VM/XA ALTCMS mm/dd/yy
■
SYNONYM SYN
CP TERM MODE VM
Ready; T=n.nn/n.nn hh:mm:ss
```

Step 13. Test the System

	Now that you have applied service to all necessary files, you must test the system to make sure that the problem has been fixed.	

1. Test the system.
2. To allow selected users to test the system, have them issue the following commands:

```
#cp link maint 490 490 rr ■
password ■
set sysname cmsdos altdos ■
set sysname cmsbam altbam ■
set sysname cmsvsam altvsam ■
set sysname cmsams altams ■
ipl altcms ■
```

	If the service performs to your satisfaction, perform Steps 14 through 19 to rebuild your system on the 190 disk. If the service fails, then you must find the problem, correct it, and retest the system.	

Step 14. Purge the Test Named Saved Systems and Saved Segments

1. Issue QUERY NSS ALL to determine the spoolids of the test named saved systems and saved segments segments you created in Step 10, Step 11, and Step 12 (ALTCMS, ALTCMSXA, ALTDSBAM, ALTDOSIN, ALTDOS, ALTBAM, ALTVSAM, ALTAMS, ALTINSHP, ALTINST, and ALTHELP).

```
query nss all ■
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMS   NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTCMSXA NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTDSBAM DCSS     MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTDOSIN DCSS     MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTDOS   DCSS     MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTBAM   DCSS     MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTVSAM  DCSS     MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTAMS   DCSS     MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTINSHP DCSS     MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTINST  DCSS     MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss ALTHELP  DCSS     MAINT
```

The spoolid is the number in the second column, under the heading "FILE."

```
Ready; T=n.nn/n.nn hh:mm:ss
```

2. Purge the named saved systems and saved segments:

```
purge nss spoolid1 ... spoolidn ■
```

You can list as many spoolids as necessary on a single PURGE NSS command.

The PURGE NSS command will take effect as soon as all users stop using the named saved systems and saved segments.

Step 15. DDR Alternate Disks to System Disks

1. Rename DMSNGP TEXT to save it. Then edit the DMSNGP ASSEMBLE file and make the following changes:
   a. Change SYSDISK = 490 to SYSDISK = 190.
   b. Change IPLADDR = 490 to IPLADDR = 190.
   c. Change HELP = 49D to HELP = 19D (or HELP = 49C to HELP = 19C).
   d. Change SYSNAME = *newname* to SYSNAME = CMS, where *newname* is the name you choose for your test system. The sample procedure in this chapter uses ALTCMS.
   e. Change INSTSEG = *newname* to SYSNAME = CMSINST, where *newname* is the name you choose for your test installation segment. The sample procedure in this chapter uses ALTINST.
   f. Change the VERSION = and INSTID = parameters to identify your new CMS system.

   These changes will cause the new CMS nucleus (with service) to be rebuilt on the 190 minidisk.
2. Establish the appropriate minidisk access order:

   ```
   setup ■
   ```

3. Reassemble the DMSNGP ASSEMBLE file:

   ```
   vmfhasm dmsngp 56643082 cms ■
   ```

4. Copy DMSNGP TEXT to the CMS alternate LOCAL1 disk (395):

   ```
   copy dmsngp text a = = b (replace ■
   ```

5. Use the DASD DUMP Restore program to copy the 490 disk to the 190 disk.

   ```
   ddr ■
   VM/XA SYSTEM PRODUCT DASD DUMP RESTORE PROGRAM
   ENTER CARD READER ADDRESS OR CONTROL STATEMENTS.
   ENTER:
   ```

   sysprint cons ■
   ENTER:

   This command tells DDR to send program messages to your console.

   input 490 *devtype* **MNT490 ■**
   ENTER:

   490 is your test CMS system disk. *devtype* is the device type of the DASD volume where 490 is located.

   output 190 *devtype* **MNT190 ■**
   ENTER:

   190 is your new CMS system disk. *devtype* is the device type of the DASD volume where 190 is located.

```
copy 000 endcyl ■
```

The value for *endcyl* depends on the device type of your 490 disk:

Device Type	*endcyl*
3350	73
3375	112
3380	71

```
DMKDDR711R Volid Read is MNT490.  Do you
           wish to continue?
yes ■
ENTER NEXT EXTENT OR NULL LINE:
■
:
END OF COPY
ENTER:
■
END OF JOB
```

You have not yet changed the label of the 190 minidisk. (You do so in substep 6.)

6. Relabel the 190 minidisk:

```
access 190 c ■
DMSACP723I C(190) R/O
DMSACC725I 190 ALSO = S DISK
```

```
format 190 c (label ■
ENTER LABEL:
mnt190 ■
Ready; T=n.nn/n.nn hh:mm:ss
release c ■
Ready; T=n.nn/n.nn hh:mm:ss
```

You **must** use the **label** option. If you don't, you erase all the files on the 190 minidisk.

7. Repeat substeps 5 on page 497 and 6 to copy the alternate HELP disks (49D, 49C, 49B...) to their corresponding current disks (19D, etc.) and to relabel the current disks as MNT19*n*. Check the minidisk definitions in your user directory to find the appropriate value for *endcyl*.
8. Erase DMSNGP TEXT from your A-disk:

```
erase dmsngp text a ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Step 16. Rebuild the Nucleus

1. Establish the appropriate minidisk access order:

   setup ■

2. Generate the new CMS nucleus:

   spool punch * class n ■
   spool prt * ■
   spool rdr class n ■

   vmfbld 56643082 cms (punch ■

   If you are processing PUT service, use **cms** as the component name.

   or

   vmfbld 56643082 corcms (punch ■

   If you are processing COR service, use **corcms** as the component name.

   ```
   DMSBLD1851I Processing CMSLOAD with
               the part handler VMFBDNUC EXEC
   LOAD LIST: $$$TLL$$ EXEC A1 (MNT191)
   ```

   This invokes the VMFBLD EXEC to build a new nucleus for CMS. The VMFBLD EXEC performs a number of system generation functions for you. For more information on this EXEC, see "VMFBLD EXEC" on page 679.

   ```
   RDR FILE fileno SENT FROM MAINT PUN AS fileno
      RECS nnnn COPY 001 N NOHOLD NOKEEP
   ```

   The system loader punches the CMS nucleus. This message tells you that the punch file is complete.

   ```
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   If you get return code 4, the reason may be that a PTF which is now being applied has another PTF, outside CMS, as a prerequisite. Check the $VMFBLD $ERRLOG file for message DMSBNC1854W.

3. Review the build exception log ($VMFBLD $ERRLOG). If necessary, correct any problems before going on.

   vmfview build ■

4. IPL your virtual reader:

   ipl 00c ■

   ```
   DMSINS327I  The installation saved segment could not be loaded.
   ```

   Informational message. The CMSINST installation segment has not been loaded and saved yet.

```
VM/XA CMS 5.5 mm/dd/yy hh:mm
■
SYNONYM SYN
CP TERM MODE VM

Ready; T=n.nn/n.nn hh:mm:ss
```

The ready message indicates that the nucleus has been loaded to your 190 minidisk successfully.

5. Query the reader to identify the CMS load map:

```
query rdr * all ■

ORIGINID FILE   CLASS RECORDS  CPY HOLD DATE  TIME     NAME     TYPE DIST
MAINT    fileno M PUN nnnnnnnn 001 NONE mm/dd hh:mm:ss              SYSPROG
```

The CMS load map is the file with a blank filename and filetype. It has approximately 8100 records. The exact size varies according to the system default language, local modifications, and VMSUP level. Note the *fileno* of this file. You will use it in your next command.

6. Receive the CMS load map that is in your reader.

```
spool prt * nohold ■
close prt * ■

setup ■
receive fileno cmsnuc map b (replace ■
CMSNUC MAP B1 created
DMSRDC738I Record length is 132 bytes
File CMSNUC MAP B received from MAINT at
  nodeid as CMSNUC MAP B
Ready; T=n.nn/n.nn hh:mm:ss
```

The CMS map is loaded onto the CMS LOCAL1 (0395) minidisk.

	Warning: From this point on, use IPL 190 instead of IPL 490 whenever you need to re-IPL CMS. The nucleus you just built is on the 190 disk; the old production nucleus (former current nucleus) is on the 490 disk.	

Step 17. Rebuild Named Saved Systems

1. Use the SAMPNSS EXEC to define segments and save your new CMS and CMSXA:

```
sampnss cms cmsxa ■
```

The SAMPNSS EXEC issues the DEFSYS command.

```
HCPNSD440I The Named Saved System (NSS) CMS was
           successfully defined in fileid fileno.
HCPNSD440I The Named Saved System (NSS) CMSXA was
           successfully defined in fileid fileno.
```

Note that the names are positional on the SAMPNSS command (the System/370 name followed by the 370-XA name).

The SAMPNSS EXEC first issues the DEFSYS command to define a skeleton system data file for CMS and for CMSXA. These messages tell you the DEFSYS commands were successful.

```
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
⋮
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss CMS      NSS      MAINT
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss CMSXA    NSS      MAINT
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

The SAMPNSS EXEC issues a QUERY NSS command. These messages show what system data files are defined.

2. Issue the QUERY NSS ALL MAP command to make sure CMS is defined properly. Check to see whether the information under BEGPAG, ENDPAG, TYPE, and CL in the response to QUERY NSS ALL MAP is the same as shown.

```
query nss all map ■
FILE FILENAME FILETYPE MINSIZE  BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
⋮
nnnn CMS      NSS      000256K  00000  0000A   EW  S  00000  OMITTED  NO
                                00020  00022   EW
                                00E00  00FFF   SR
nnnn CMSXA    NSS      000256K  00000  0000A   EW  S  00000  OMITTED  NO
                                00020  00022   EW
                                00E00  00FFF   SR
⋮
```

3. Set your machine mode to XA, load CMS (IPL 190), and save the CMS system.

```
set machine 370 ■
```

```
ipl 190 clear parm savesys cms ■
```

Load 190 with the option to save the system under the name CMS.

```
HCPNSS440I The named saved system CMS was
           successfully saved in fileid fileno.
VM/XA CMS 5.5 mm/dd/yy hh:mm
```

■
Ready; T=*n.nn/n.nn hh:mm:ss*

Press **ENTER** to initialize CMS.

4. Set your machine mode to XA, load CMS (IPL 190), and save the CMSXA system.

set machine xa ■

ipl 190 clear parm savesys cmsxa ■

Load 190 with the option to save the system under the name CMSXA.

HCPNSS440I The named saved system CMSXA was
successfully saved in fileid *fileno.*
VM/XA CMS 5.5 *mm/dd/yy hh:mm*

■
Ready; T=*n.nn/n.nn hh:mm:ss*

Press **ENTER** to initialize CMSXA.

5. Set your machine mode to 370 (or XA), redefine your virtual storage to 2 megabytes, and load the CMS (or CMSXA) named saved system. The example below shows loading CMS in System/370 mode:

set machine 370 ■
define storage 2m ■

STORAGE = 2M
STORAGE CLEARED - SYSTEM RESET

Defining storage causes a system reset.

ipl cms ■
VM/XA CMS 5.5 *mm/dd/yy hh:mm*

■
Ready; T=*n.nn/n.nn hh:mm:ss*

After receiving the version identification, press **ENTER** to complete CMS or CMSXA initialization.

6. Check the CMS status by issuing QUERY NSS ALL and QUERY NSS ALL MAP. Check to see whether the information under FILENAME and FILETYPE in the response to QUERY NSS ALL is the same as shown. Also, check whether the information under BEGPAG, ENDPAG, TYPE, and CL in the response to QUERY NSS ALL MAP is the same as shown. The system data file should now have class A (rather than S) and have one user (MAINT).

```
query nss all ■
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
⋮
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMS      NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMSXA    NSS      MAINT
⋮
Ready; T=n.nn/n.nn hh:mm:ss
query nss all map ■
FILE FILENAME FILETYPE MINSIZE  BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
⋮
nnnn CMS      NSS      000256K  00000  0000A   EW  A  00001  OMITTED  NO
                                00020  00022   EW
                                00E00  00FFF   SR
nnnn CMSXA    NSS      000256K  00000  0000A   EW  A  00000  OMITTED  NO
                                00020  00022   EW
                                00E00  00FFF   SR
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

Step 18. Reinstall the CMSDOS, CMSBAM, CMSVSAM, and CMSAMS Saved Segments

	Warning: Do not skip this step even if you have not changed CMSDOS, CMSBAM, CMSVSAM, or CMSAMS. **Warning: The CMSDOS and CMSBAM segments must be installed before you install the CMSVSAM and CMSAMS segments.**	

1. Define your storage as 16 megabytes and IPL your new CMS system.

```
define storage 16m ■
STORAGE = 0016M
STORAGE CLEARED - SYSTEM RESET
ipl cms ■


VM/XA CMS 5.5 mm/dd/yy hh:mm
** DO NOT press ENTER! **
```

```
access (noprof ■
```

This command suppresses execution of MAINT's PROFILE EXEC.

2. Establish the appropriate minidisk access order:

```
setup ■
```

3. Issue:

```
set emsg on ■
Ready; T=n.nn/n.nn hh:mm:ss
```

You want to see any and all error messages during execution of the installation EXEC.

4. Define a segment into which the DOSGEN EXEC can load CMSDOS.

```
defseg dosinst 900-90f sr ■


HCPNSD440I  Saved segment DOSINST was successfully
              defined in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
```

This command defines a 1MB segment for DOSINST starting at 900000. You may place this segment anywhere below the segment spaces defined for CMSDOS, CMSBAM, CMSVSAM, and CMSAMS.

5. Invoke the DOSGEN EXEC with a load address and the name DOSINST. The load address used for DOSINST in this example is 900000.

```
dosgen 900000 dosinst ■
```

```
HCPNSS440I Saved segment DOSINST was successfully
             saved in fileid fileno
PRT FILE fileno SENT FROM MAINT PRT AS fileno
  RECS nnnn COPY 001 A NOHOLD NOKEEP
DMSWGN715I DOSGEN COMPLETE
Ready; T=n.nn/n.nn hh:mm:ss
```

The load address and name are those recommended for the DOSINST segment. Error messages for the DOSGEN EXEC are listed on page 486.

6. Re-IPL your new CMS system:

```
ipl cms ■
VM/XA CMS 5.5 mm/dd/yy hh:mm
■
SYNONYM SYN
CP TERM MODE VM
Ready; T=n.nn/n.nn hh:mm:ss
```

7. Establish the appropriate minidisk access order:

```
setup ■
```

8. Define CMSDOS, CMSBAM, CMSAMS, and CMSVSAM:

```
set sysname cmsdos dosinst ■
Ready; T=n.nn/n.nn hh:mm:ss
sampnss cmsdos ■
HCPNSD440I Saved segment CMSDOS was successfully
             defined in fileid fileno
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMS      NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMSXA    NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss GCS      NSS      MAINT
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss DOSINST  DCSS     MAINT
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss DOSBAM   DCSS     MAINT
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss CMSDOS   DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss

sampnss cmsbam ■
HCPNSD440I Saved segment CMSBAM was successfully
             defined in fileid fileno
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMS      NSS      MAINT
:
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss CMSBAM   DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss

sampnss cmsams ■
HCPNSD440I Saved segment CMSAMS was successfully
             defined in fileid fileno
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMS      NSS      MAINT
:
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss CMSAMS   DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

```
sampnss cmsvsam ■
HCPNSD440I Saved segment CMSVSAM was successfully
            defined in fileid fileno
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMS      NSS      MAINT
:
*NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss CMSVSAM  DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

9. Invoke the DOSGEN EXEC with a load address and the name CMSDOS. The load address set up by SAMPNSS is B00000 for the name CMSDOS.

```
dosgen b00000 cmsdos ■
```

The load address and name are those recommended for the CMSDOS segment. Error messages for the DOSGEN EXEC are listed on page 486.

```
HCPNSS440I Saved segment CMSDOS was successfully
            saved in fileid fileno
PRT FILE fileno SENT FROM MAINT PRT AS fileno
  RECS nnnn COPY 001 A NOHOLD NOKEEP
DMSGEN715I DOSGEN COMPLETE
Ready; T=n.nn/n.nn hh:mm:ss
```

10. To save the load map, rename it and copy it to the 193 minidisk (K).

```
copy load map a cmsdos segmap k (replace ■
Ready; T=n.nn/n.nn hh:mm:ss
```

11. Replace the name CMSBAM in the CMS SYSNAME table with any name that is NOT a name previously used as a segment name.

```
set sysname cmsbam sysname ■
Ready; T=n.nn/n.nn hh:mm:ss
```

The SET SYSNAME command enters an alternate name for CMSBAM in the SYSNAME table. Choose a name that was **not** used previously as a segment name. This command does not actually change the saved segment's name, but keeps CP from finding and IPLing the segment at the wrong time.

12. Place your CMS virtual machine in a CMS/DOS environment, then invoke SAMGEN to load the CMSBAM segment. You must give the EXEC an address at which to load the CMSBAM segment; this address also must match the address in the skeleton CMSBAM segment that you defined beforehand.

```
set dos on ■
```

This command places your virtual machine in a CMS/DOS environment.

```
DMSSET400I SYSTEM sysname DOES NOT EXIST
DMSSET1101I 100K DOS partition defined
             at hexadecimal location 020000.
Ready; T=n.nn/n.nn hh:mm:ss
samgen ■
DMSSGN363R ENTER LOCATION WHERE CMSBAM
WILL BE LOADED AND SAVED:
```

sysname is the name you chose in substep 11.

```
b10000 ■
DMSSGN364I FETCHING CMSBAM...
DMSFET710I PHASE DMSVBM ENTRY POINT AT LOCATION B100C0.
DMSSGN366R ENTER NAME OF SYSTEM TO BE SAVED:
cmsbam ■
```

```
HCPNSS440I Saved segment CMSBAM was successfully
             saved in fileid fileno
DMSSGN365I SYSTEM CMSBAM SAVED
Ready; T=n.nn/n.nn hh:mm:ss
```

The messages indicate that the segment has been loaded and saved.

13. Access the VSAM product disk (the same disk you accessed in substep 13 on page 487 of Step 11) as your A-minidisk.

```
access vdevno a ■
Ready; T=n.nn/n.nn hh:mm:ss
```

14. Invoke VSAMGEN EXEC:

```
vsamgen ■
SELECT ONE OF THE FOLLOWING FUNCTIONS BY ENTERING THE NUMBER:

 1. INSTALL AMS            (READ VSAM PRODUCT TAPE, BUILD DOSLIB, CREATE SEGMENT)
 2. INSTALL VSAM           (READ VSAM PRODUCT TAPE, BUILD DOSLIB, CREATE SEGMENT)
 3. INSTALL VSAM AND AMS   (READ VSAM PRODUCT TAPE, BUILD DOSLIB, CREATE SEGMENT)
 4. BUILD  AMS             (BUILD DOSLIB, CREATE SEGMENT)
 5. BUILD  VSAM            (BUILD DOSLIB, CREATE SEGMENT)
 6. BUILD  VSAM AND AMS    (BUILD DOSLIB, CREATE SEGMENT)
 7. RESTART AMS            (CREATE SEGMENT)
 8. RESTART VSAM           (CREATE SEGMENT)
 9. RESTART VSAM AND AMS   (CREATE SEGMENT)
10. QUIT                   (EXIT VSAMGEN EXECUTION)

       ENTER RESPONSE...
```

```
6 ■
```

Choosing option 6 tells the EXEC to create both CMSVSAM and CMSAMS segments as new segments, without reading the text files from the VSAM product tape. If you read in new files from the VSAM product tape in substep 14 on page 488 of Step 11, you will use those files now.

Messages

While VSAMGEN is processing, you may receive error and information messages. These messages are self-explanatory. Messages labeled 2101I are information messages from the linkage editor and may be ignored.

Respond to the EXEC messages as they appear:

```
DMSVGN362I LINK-EDITING CMSVSAM ...
DMSVGN363I CMSVSAM DOSLIB CREATED ON DISK 'A'.
DMSVGN370R ENTER 'GO' IF SAVED SYSTEM IS TO BE CREATED,
             OTHERWISE ENTER 'QUIT'.
go ■
DMSVGN363R ENTER LOCATION WHERE CMSVSAM SHARED WILL BE
          LOADED AND SAVED:


ba0000 ■


DMSVGN363R ENTER LOCATION WHERE CMSVSAM NONSHARED WILL BE
          LOADED AND SAVED:
a30000 ■


DMSVGN364I FETCHING CMSVSAM ...
DMSFET710I PHASE DMSVVS ENTRY POINT AT LOCATION nnnnnn
DSMVGN371R CMSVSAM IS LOADED, IF ZAPS ARE TO BE APPLIED GO INTO
        'CP' MODE, APPLY THE ZAPS AND THEN REPLY 'GO'
go ■
DMSVGN366R ENTER NAME OF SYSTEM TO BE SAVED:


cmsvsam ■
```

This name is the default name with which the segment was defined.

```
DMSVGN365I SYSTEM CMSVSAM SAVED.

DMSVGN368R ERASE CMSVSAM DOSLIB ? ... ENTER 'YES' OR 'NO':
no ■


DMSVGN362I LINK-EDITING CMSAMS ...
DMSVGN363I CMSAMS DOSLIB CREATED ON DISK 'A'.
DMSVGN370R ENTER 'GO' IF SAVED SYSTEM IS TO BE CREATED,
             OTHERWISE ENTER 'QUIT'.
go ■
DMSVGN363R ENTER LOCATION WHERE CMSAMS SHARED WILL BE
          LOADED AND SAVED:
```

```
b40000 ■                                          This is the recommended location.
DMSVGN363R ENTER LOCATION WHERE CMSAMS NONSHARED WILL BE
          LOADED AND SAVED:
a00000 ■
DMSVGN364I FETCHING CMSAMS ...
DMSFET710I PHASE DMSVAS ENTRY POINT AT LOCATION nnnnnn
DMSFET710I PHASE DMSVAN ENTRY POINT AT LOCATION nnnnnn
DMSFET710I PHASE DMSVAX ENTRY POINT AT LOCATION nnnnnn
DMSVGN371R CMSAMS IS LOADED, IF ZAPS ARE TO BE APPLIED GO INTO
        'CP' MODE, APPLY THE ZAPS AND THEN REPLY 'GO'
go ■
DMSVGN366R ENTER NAME OF SYSTEM TO BE SAVED:


cmsams ■                                          This name is the default name used for
                                                  defining the segment.


DMSVGN365I SYSTEM CMSAMS SAVED.

DMSVGN368R ERASE CMSAMS DOSLIB ? ... ENTER 'YES' OR 'NO':
no ■
Ready; T=n.nn/n.nn hh:mm:ss
```

15. You may now purge the DOSINST named saved segment:

```
query nss all ■
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMS      NSS      MAINT
⋮
*NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss DOSINST  DCSS     MAINT
⋮
Ready; T=n.nn/n.nn hh:mm:ss


purge nss fileno ■                                fileno is the file identifier for DOSINST.
No files purged
0001 file pending purged
Ready; T=n.nn/n.nn hh:mm:ss
```

16. Set DOS off:

```
set dos off ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Step 19. Reinstall the CMSINST and HELP Saved Segments

	Warning: Do not skip this step even if you have not changed CMSINST or HELP.	

1. Make sure that you have the right NAMESAVE segments in your user directory:
   - You must have the NAMESAVE HELP statement in your user directory in order to save the HELP file directory information in a saved segment.
   - If the HELP segment is defined as a member of a segment space, you must have a NAMESAVE statement (for example, NAMESAVE INSTHELP) in the directory entry for the user who invokes SAVEFD. SAVEFD specifies the name of the segment space.
2. Log onto MAINT (unless you are continuing from the previous step).
3. Check your virtual storage. If it is less than 16MB, issue the following:

```
define storage 16m ■
STORAGE CLEARED - SYSTEM RESET
STORAGE = 0016M
```

4. IPL your new CMS system.

```
ipl cms ■
VM/XA CMS 5.5 mm/dd/yy hh:mm

■
SYNONYM SYN
CP TERM MODE VM
Ready; T=n.nn/n.nn hh:mm:ss
```

   Press **ENTER** to complete the CMS initialization.

5. Establish the appropriate minidisk access order:

```
setup ■
```

6. Purge your old CMSINST segment:

```
segment purge cmsinst ■
Ready; T=n.nn/n.nn hh:mm:ss
```

7. Define a segment for CMSINST:

```
sampnss cmsinst ■
HCPNSD440I Saved segment CMSINST was successfully
           defined in fileid fileno
OWNERID FILE TYPE CL RECS DATE  TIME  FILENAME FILETYPE ORIGINID
*NSS    nnnn NSS  A  nnnn mm:dd hh:mm CMS      NSS      MAINT
:
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm CMSINST  DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

The EXEC issues a QUERY command that displays information about the segments defined for named saved systems.

8. Define the HELP segment:

```
sampnss help ■
HCPNSD440I Saved segment HELP was successfully
           defined in fileid fileno
OWNERID FILE TYPE CL RECS DATE  TIME  FILENAME FILETYPE ORIGINID
*NSS    nnnn NSS  A  nnnn mm:dd hh:mm CMS      NSS      MAINT
:
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm CMSINST  DCSS     MAINT
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm HELP     DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

9. Save both segments:

```
saveseg cmsinst ■
```

Each member saved segment defined by SAMPNSS must be saved separately.

```
HCPNSS440I Saved segment CMSINST was successfully
           saved in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
saveseg help ■
HCPNSS440I Saved segment HELP was successfully
           saved in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
```

10. Redefine each segment to create the skeleton segments:

```
sampnss cmsinst ■
HCPNSD440I Saved segment CMSINST was successfully
            defined in fileid fileno
OWNERID FILE TYPE CL RECS DATE  TIME  FILENAME FILETYPE ORIGINID
*NSS    nnnn NSS  A  nnnn mm:dd hh:mm CMS      NSS      MAINT
⋮
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm CMSINST  DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
sampnss help ■
HCPNSD440I Saved segment HELP was successfully
            defined in fileid fileno
OWNERID FILE TYPE CL RECS DATE  TIME  FILENAME FILETYPE ORIGINID
*NSS    nnnn NSS  A  nnnn mm:dd hh:mm CMS      NSS      MAINT
⋮
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm CMSINST  DCSS     MAINT
*NSS    nnnn NSS  S  nnnn mm:dd hh:mm HELP     DCSS     MAINT
Ready; T=n.nn/n.nn hh:mm:ss
```

11. You may have created a load list for the DCSSGEN command during the installation process. The one created in the sample installation process was called INSTLIST FILE. It should be on your A-disk (191). If it is not there, you probably used the IBM-supplied load list. This load list is called CMSINST EXECLIST. It should be on your K-disk (193). Using one of these load lists, issue the DCSSGEN command as follows:

```
dcssgen fn ft fm cmsinst ■
```

fn ft fm is the fileid of the file that contains the list of EXECs and Editor macros to be loaded into the segment.

CMSINST is the name of the segment. CMSINST is the default name used for defining the segment and for specifying the segment name in DMSNGP or in response to the installation questions. If you do not specify a segment name, the default name is CMSINST.

```
HCPNSS440I Saved segment CMSINST was successfully
            saved in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
```

Note: When you built your CMS nucleus, if you indicated in the DMSNGP file (USEINST = YES) or in answer to the DMSINQ296R prompt that you wanted to use the installation segment (the default is YES), then this segment is used each time a user IPLs CMS. If you indicated earlier that you did not want to use the installation segment but now want to use it, you must modify the DMSNGP file to indicate that the segment should be used, then assemble the modified DMSNGP and rebuild CMS.

12. Define your virtual storage as **less** than the address at which the HELP segment is to be loaded. For example, if the HELP segment is defined at X'C50000', define your storage as 12MB.

```
define storage 12m ■
STORAGE CLEARED - SYSTEM RESET
STORAGE = 12M
```

13. Re-IPL your new CMS system.

```
ipl cms ■


VM/XA CMS 5.5 mm/dd/yy hh:mm
■
SYNONYM SYN
CP TERM MODE VM
Ready; T=n.nn/n.nn hh:mm:ss
```

14. Establish the appropriate minidisk access order:

```
setup ■
```

15. Issue the following commands to initialize and save the segment:

```
savefd init vaddr label help ■
Ready; T=n.nn/n.nn hh:mm:ss
```

vaddr is 19D for mixed-case American English, 19C for uppercase American English.

label is the CMS label assigned to the disk. (The labels in the sample directory are MNT19D and MNT19C.)

```
savefd save vaddr label help ■
DMSACP723I Z(19D) R/O
HCPNSS440I Saved segment HELP was successfully
            saved in fileid fileno
Ready; T=n.nn/n.nn hh:mm:ss
```

16. For more information about using the SAVEFD command to save minidisk file directory information in a saved segment, refer to *VM/XA SP CMS Command Reference*. For more information about saved segments, refer to *VM/XA SP Guide to Saved Segments*.

17. Verify that the CMSINST and CMSHELP segments are defined correctly:

```
query nss all ■
OWNERID FILE   TYPE CL RECS DATE  TIME     FILENAME    FILETYPE ORIGINID
⋮
*NSS    fileno NSS  A  nnnn mm/dd hh:mm:ss INSTHELP    NSS      MAINT
*NSS    fileno NSS  A  nnnn mm/dd hh:mm:ss CMSINST     NSS      MAINT
*NSS    fileno NSS  A  nnnn mm/dd hh:mm:ss HELP        NSS      MAINT
```

18. Redefine your storage before you continue:

```
define storage 16m ■
STORAGE = 16M
Storage cleared - system reset
ipl cms ■
VM/XA CMS 5.5 mm/dd/yy hh:mm
■
SYNONYM SYN
CP TERM MODE VM
Ready; T=n.nn/n.nn hh:mm:ss
```

Step 20. Back Up the Named Saved Systems

1. Use the procedure in Step 29 of Chapter 2, "Installing VM/XA System Product Release 2 with the Starter System (First Level)" to back up the new CMS and CMSXA named saved systems.

MAP 0013: What to Do Next

001

You have finished the service procedure for CMS.

Are you also servicing CP?
Yes **No**

002

– Stop here.

003

– Go to Chapter 14, "Rebuilding CP after Applying Service" on page 515.

Chapter 14. Rebuilding CP after Applying Service

This chapter describes:

- A step-by-step procedure for rebuilding CP after applying program update service, corrective service, or local service.

MAP 0014: Should You Be Doing This Now?

001

Are you also servicing CMS?
Yes **No**

002

– Continue with this chapter.

003

Have you built CMS?
Yes **No**

004

– Go to Chapter 13, "Rebuilding CMS after Applying Service" on page 457.

005

– Continue with this chapter.

Step 1. Preparation

1. Establish the appropriate minidisk access order:

```
setup ■
Ready; T=n.nn/n.nn hh:mm:ss
```

2. Use the DASD DUMP Restore program to copy the current CMS BUILD1 disk to its corresponding alternate disk.

```
ddr ■
VM/XA SYSTEM PRODUCT DASD DUMP RESTORE PROGRAM
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS.
ENTER:
```

```
sysprint cons ■
ENTER:
```

This command tells DDR to send program messages to your console.

```
input 190 devtype MNT190 ■
ENTER:
```

190 is your current CMS system disk. *devtype* is the device type of the DASD volume where 190 is located.

```
output 490 devtype MNT490 ■
ENTER:
```

490 is your alternate CMS system disk. *devtype* is the device type of the DASD volume where 490 is located.

```
copy 000 endcyl ■
```

The value for *endcyl* depends on the device type of your 190 disk:

Device Type	*endcyl*
3350	73
3375	112
3380	71

```
DMKDDR711R Volid Read is MNT190.  Do you
           wish to continue?
yes ■
ENTER NEXT EXTENT OR NULL LINE:
■
⋮
END OF COPY
ENTER:
■
END OF JOB
```

You have not yet changed the label of the 490 minidisk. (You will do so in substep 3 on page 517.)

3. Relabel the 490 minidisk:

```
access 490 t ■
Ready; T=n.nn/n.nn hh:mm:ss
format 490 t (label ■
ENTER LABEL:
mnt490 ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Access 490 *after* the system disk. You *must* use the **label** option. If you don't, you will erase all the files on the 490 minidisk.

Step 2. Build a New CP Macro Library

1. Establish the appropriate minidisk access order:

```
setup ■
Ready; T=n.nn/n.nn hh:mm:ss
vmfsetup 56643082 cp (bld ■
DMSWSU1900W The existing 56643082 $SETUP A1 file
              has been refreshed.  You might want to
              check your access order when done.
Ready(0004); T=n.nn/n.nn hh:mm:ss
```

Program Update Service Only

2. If you are processing program update service, erase CPNEW MACLIB if it exists. **Do not erase CPNEW MACLIB if you are processing corrective service or local service.**

```
erase cpnew maclib fm ■
Ready; T=n.nn/n.nn hh:mm:ss
```

End of Program Update Service Only

	Do the rest of this step if: • **You have not already done it as part of the CMS service process *and*** • **One of these conditions applies:** – **You are processing program update service and have local modifications to CP macros or control blocks** – **You are processing corrective service** – **You are processing local service.**	

3. Access 295 as A:

```
access 295 a ■
```

4. Find out which macros and control blocks have been updated by IBM. First list all the CP updates you received from IBM:

```
listfile * H*HP fm (EXEC ■
rename cms EXEC a cpupdate = = ■
xedit cpupdate EXEC ■
sort * 1 50 ■
file ■
```

The updates are on the CP alternate LOCAL1 (295) disk if you are applying corrective service or on the CP alternate DELTA (594) disk if you are applying program update service. You may also have updates on other disks in the LOCAL1 string.

The **EXEC** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

5. Find out which of the updates you received affect macros and control blocks. At the same time, note where those macros and control blocks are. The source files for macros and control blocks should be on the CP BASE2 disk (394). For each update listed in CPUPDATE EXEC, issue:

```
listfile fn macro * ■
Ready; T=n.nn/n.nn hh:mm:ss
listfile fn copy * ■
Ready; T=n.nn/n.nn hh:mm:ss
```

If the update affects a macro or control block file, that file will be listed.

6. Create or edit an EXEC called CPNEW EXEC to generate a macro library for updated macros. It should have the same format as HCPXA1 EXEC, and should list:
   - For program update service, only those macros and control blocks for which you have received service and for which you also have local modifications
   - For corrective service, all macros and control blocks for which you have received service, plus any for which you have local modifications
   - For local service, all macros and control blocks for which you have local modifications.

   If you have no macros or control block files that must be listed in CPNEW EXEC, go to substep 12 on page 520.

   a. If you have not already created CPNEW EXEC, issue:

```
copy hcpxa1 EXEC fm cpnew = a (recfm f lrecl 80 ■
```

   b. If you already have a copy of CPNEW EXEC, issue:

```
copy cpnew EXEC fm = = a (recfm f lrecl 80 ■
```

   Now edit CPNEW EXEC, adding all necessary macro and control block names. If you created CPNEW EXEC by copying HCPXA1 EXEC, use the contents of HCPXA1 EXEC only as a guide to the format of the macro entries. After you have added the macros and control block names you need, in the proper format, delete the macro names that came from HCPXA1 EXEC.

```
xedit cpnew EXEC a ■
⋮
file ■
```

7. You now need to copy every macro and control block file listed in CPNEW EXEC to your A-disk.

```
copy fn macro fm = = a (olddate replace ■
copy fn copy fm = = a (olddate replace ■
```

8. Generate the new macro library by using the VMFMAC EXEC:

```
vmfmac cpnew hcpxa ■
DMSUPD178I Updating macroname MACRO A1
DMSUPD178I Applying macroname update A1
macroname MACRO ADDED
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

   Note: An alternate method to using the VMFMAC EXEC is to update the control blocks and macros by using the UPDATE module and issuing MACLIB DELETE and then MACLIB ADD commands for the MACLIB containing the changed macro or control block.

   See *VM/XA SP CMS Command Reference* for details on the MACLIB command.

9. The new macro library is now on your A-disk (295). You can now erase CPNEW COPY (a file created by VMFMAC) from the 295 disk.

```
erase cpnew copy a ■
```

10. Add the new macro library name to the TEXT MACS card in the HCPXA CNTRL file. Make it the first macro library listed so that you use the updated macros.

```
xedit hcpxa cntrl ■
locate/TEXT MACS ■
TEXT MACS HCPXA1 ...
change/MACS/MACS CPNEW/ ■
TEXT MACS CPNEW HCPXA1 ...
file ■
```

11. If you want to make the new macro library available to general users, copy it to the alternate CMS BUILD1 disk (490):

```
copy cpnew maclib a = = r2 (olddate replace ■
```

12. Reaccess 295 as B and 191 as A:

```
access 295 b ■
access 191 a ■
```

	If you have any image library updates, refer to *VM/XA SP Planning and Administration.*	

Step 3. Update the CP Message Repository

	Do this step if: • **You are processing program update service and have local modifications to the CP message repository** • **You are processing corrective service and have local modifications to the CP message repository** • **You are processing local service.**	

1. Determine your system default national language:

   ```
   query lang ■
   ```

   langid — See Table 5 on page 317 to identify the language corresponding to *langid*. Note the country code for that language.

2. Invoke the VMFNLS EXEC to update the CP message repository:

   ```
   vmfnls hcpmesy repos 56643082 cp ■
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   y is the country code for your system national language.

3. If you have updates to the message repositories for any other national languages installed on your system, repeat substep 2 for each language.

Step 4. Assemble the Changed ASSEMBLE Files

1. Establish the correct minidisk access order:

   setup ■
   vmfsetup 56643082 cp (all ■

2. Determine which ASSEMBLE files you must reassemble:

   - Although you will not receive any updates affecting HCPRIO ASSEMBLE or HCPSYS ASSEMBLE, you must reassemble these files.
   - Review the apply exception log ($VMFAPP $ERRLOG) for the names of ASSEMBLE files to which you have applied service and for which you also have local updates. You decided in Chapter 11, "Applying Program Update Service or Corrective Service to CMS" on page 449 which of these local updates you wish to keep.

   vmfview apply ■

   To find these ASSEMBLE files, search for all instances of message DMSxxx1885W. This message lists all the MACRO, COPY, EXEC, and ASSEMBLE files for which you have local updates. You must reassemble any such ASSEMBLE files to pick up the local updates.

	If you must reassemble any ASSEMBLE files, continue with substep 3. Otherwise, go to "Step 5. Build the CP Nucleus" on page 523.	

3. Copy and unpack each ASSEMBLE file to be reassembled. You must do this because the UPDATE module requires unpacked files, and they are packed on the 394 disk, where they are stored.

   copy hcp*xxx* assemble * = = a (unpack olddate ■

   Substitute the last three characters of the filename of the ASSEMBLE file for *xxx*.

4. Use the VMFHASM EXEC to update and assemble all the files that need reassembling:

   vmfhasm hcp*xxx* 56643082 cp ■
   Ready; T=*n.nn/n.nn hh:mm:ss*

   Substitute the last three characters of the filename of the ASSEMBLE file for *xxx*.

   The text file created will be placed on the A-disk.

5. Copy all the text files from the A-disk to the alternate LOCAL1 (295) disk:

   copy hcp*xxx* {**text**|*xxxnnnnn*} **a = = b** ■

6. If you reassembled any files that are not in the CP load list, copy the text decks to the alternate LOCAL1 disk again, giving the copy the filetype TEXT:

   copy hcp*xxx* {**text**|*xxxnnnnn*} **a = text b** ■

7. Erase all the ASSEMBLE files and text files from your A-disk:

   erase hcp*xxx* assemble a ■
   erase hcp*xxx* {**text**|*xxxnnnnn*} **a** ■

8. Repeat substeps 3 through 7 for each file that must be reassembled.

Step 5. Build the CP Nucleus

	If you have received corrective service for, or have local modifications to, HCPLDL ASSEMBLE or the HCPMDLAT macro, you must issue UTILITY CPLOAD to rebuild the CPLOAD EXEC before you build the nucleus. See Step 8 for more details.	

1. Make a backup copy of your system, using the DASD dump restore program.
2. Define storage as 16 megabytes and IPL your system disk:

```
define storage 16m ■
STORAGE =      16M


STORAGE CLEARED - SYSTEM RESET


ipl sysaddr ■
VM/XA CMS 5.5 mm/dd/yy hh:mm


■
Ready; T=n.nn/n.nn hh:mm:ss
```

Defining storage causes a system reset.

sysaddr is your system disk address, either 190 or 490.

After receiving the version identification, press **ENTER**

3. Establish the appropriate minidisk access order:

```
setup ■
```

4. Invoke the VMFBLD EXEC to build a new CP nucleus.

```
spool punch * class n ■
spool prt * ■
spool rdr class n ■
spool rdr hold ■
vmfbld 56643082 cp (punch ■

   or

vmfbld 56643082 corcp (punch ■
DMSBLD1851I Processing CPLOAD with the part
              handler VMFBDNUC EXEC
LOAD LIST: $$$TLL$$ EXEC A1 (MNT191)
```

If you are processing PUT service, use **cp** as the component name.

If you are processing COR service, use **corcp** as the component name.

(To ensure that text decks are copied to the correct disk during VMFBLD execution using VMFBDCPY, the component name for CP corrective service is now **corcp**. This causes the corrective service override to be applied to the 56643082 $PPF and the DELTA disk string to be redefined).

```
RDR FILE fileno SENT FROM MAINT PUN AS fileno RECS nnnn COPY 001 N NOHOLD NOKEEP
```

The system loader punches the CP nucleus. This message tells you that the punch file is complete.

```
Ready; T=n.nn/n.nn hh:mm:ss
```

If you get return code 4, the reason may be that a PTF which is now being applied has another PTF, outside CP, as a prerequisite. Check the $VMFBLD $ERRLOG file for message DMSBNC1854W.

5. Review the build exception log ($VMFBLD $ERRLOG). If necessary, correct any problems before going on.

```
vmfview build ■
```

6. Make sure that the files created in the previous substep are in MAINT's virtual reader.

```
query rdr maint all class n ■
```

The ALL operand asks for a display of all information about the reader files.

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME     TYPE     DIST
MAINT    nnnn N PUN nnnnnnnn 001 USER mm/dd hh:mm:ss $$$TLL$$ IPL      SYSPROG
Ready(0004); T=n.nn/n.nn hh:mm:ss
```

	Perform the next two substeps only if you are generating a CP nucleus with a preferred virtual machine.	

7. Precautions must be taken if the new CP nucleus is to contain a V=R area (including V=F storage) but you are not ready to load (IPL) the new nucleus. If the V=R guest is running a job, then let it finish its present work and do not allow more work to begin. The reason for this precaution is that V=R recovery may fail in the event of a system restart.
8. Find out how much virtual storage MAINT has and define more if necessary. If you are generating a CP nucleus with a preferred virtual machine, MAINT's virtual storage must be at least 4 megabytes greater than that specified by the VRSIZE operand plus that specified by the VRFREE operand in the SYSSTORE macro instruction in HCPSYS ASSEMBLE.

```
vmfsetup 56643082 cp (all ■
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

```
xedit hcpsys assemble ■
⋮
locate /VRSIZE ■
qquit ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Check the size of the VRSIZE and VRFREE operands.

```
query virtual storage ■
STORAGE = nnnnM
define storage nnnnm ■
```

```
STORAGE = nnnnM
STORAGE CLEARED - SYSTEM RESET
```

You will re-IPL CMS in substep 14 on page 526, so you do not have to do it now.

9. If you are using the starter system defaults, issue the following commands to make sure that the new nucleus does not overlay the production nucleus on the system residence volume while your system is in production.

```
set machine 370 ■
Ready; T=n.nn/n.nn hh:mm:ss
```

```
cpformat 423 xasres 000-endcyl ■
CPFORMAT:
FORMAT WILL ERASE CYLINDERS 0000-0009 ON DISK 423
DO YOU WANT TO CONTINUE ? (YES | NO)
yes ■
FORMAT STARTED
FORMAT COMPLETE
CURRENT ALLOCATION
TYPE CYLINDERS
.....
PERM 0000-endcyl
ENTER ALLOCATION DATA
TYPE CYLINDERS
.....
```

CPFORMAT the 423 minidisk with label XASRES.

The value of *endcyl* depends on two things: the DASD type and whether you are are using the 423 minidisk defined in the sample directory, or a full-pack minidisk that you have defined on a second XASRES volume.

Device Type	*endcyl* for Sample 423	*endcyl* for Full-Pack 423
3350	0012	0554
3375	0015	0958
3380	0009	0884
3380-E4	0009	1769
3380-K	0009	2654

```
end ■
CURRENT ALLOCATION
TYPE CYLINDERS
.....
PERM 0000-endcyl
ALLOCATE COMPLETE
VOLUME LABEL IS NOW 'XASRES'
HCPFAM3841 CPFORMAT COMPLETE
SYSTEM RESET
```

Allocate the whole area as PERM.

```
define 123 nnn ■
DASD nnn DEFINED
Ready; T=n.nn/n.nn hh:mm:ss
```

nnn is any address you are not using.

```
define 423 123 ■
DASD 0123 DEFINED
Ready; T=n.nn/n.nn hh:mm:ss
```

Define the 423 minidisk as virtual 123. This is the default address for the system residence volume.

```
define nnn 423 ■
DASD 0423 DEFINED
Ready; T=n.nn/n.nn hh:mm:ss
```

Define the a backup pack for the system residence volume. This is your old 123 minidisk.

10. You must perform this substep if you redefined MAINT's virtual storage as greater than 16 megabytes.

```
set machine xa ■
```

Storage addresses greater than 16 megabytes require 370/XA architecture.

11. Load (IPL) MAINT's virtual reader.

```
ipl 00c ■
```

```
MSG FROM MAINT   :
  HCPGEN9010W NUCLEUS LOADED ON XASRES
  CP ENTERED; DISABLED WAIT
  PSW 000A0000 00009010
```

The 9010 disabled wait state indicates that the CP nucleus is successfully loaded on the system residence device. If you receive a different disabled wait state code, see *VM/XA SP System Messages and Codes Reference.*

If you entered a disabled wait state with a code 8028, you did not observe the warning notes dealing with virtual storage size in the substeps above.

12. You have now loaded the nucleus to MAINT's 423 disk. (This is also the test 123 minidisk, labeled MNT123.) If you are using the starter system defaults, this is a holding area for the new nucleus.

13. Issue the following commands to review the CP nucleus map for errors.

```
set machine 370 ■
define storage 16m ■
```

14. Save the CP load map.

    a. The load map has been spooled to MAINT's virtual printer. To save the load map on MAINT's LOCAL1 (0295) minidisk:

```
spool prt nohold ■
close prt ■
```

```
ipl sysaddr ■
```

sysaddr is your CMS system disk, either 190 or 490.

    b. When VM READ appears in the corner of your screen, continue:

```
■
CMS
setup ■
Ready; T=n.nn/n.nn hh:mm:ss
access 295 b ■
Ready; T=n.nn/n.nn hh:mm:ss
query rdr * all ■
```

```
OWNERID FILE   CLASS RECORDS  CPY HOLD USERFORM OPERFORM KEEP MSG
MAINT   fileno A PRT nnnnnnnn 001 NONE STANDARD STANDARD OFF  OFF
```

Note the *fileno* in this message. You will use it in your next command.

```
receive fileno cpnuc map b (replace ■
Ready; T=n.nn/n.nn hh:mm:ss
```

This command saves the load map on MAINT's LOCAL1 (0295) minidisk.

c. To print a copy of the load map for the CP nucleus:

```
print cpnuc map b ■
Ready; T=n.nn/n.nn hh:mm:ss
```

The load map is printed on your system printer.

15. Redefine your nucleus areas at their original addresses.

```
define 423 nnn ■
DASD nnn DEFINED
Ready; T=n.nn/n.nn hh:mm:ss
define 123 423 ■
DASD 0423 DEFINED
Ready; T=n.nn/n.nn hh:mm:ss
define nnn 123 ■
DASD 0123 DEFINED
Ready; T=n.nn/n.nn hh:mm:ss
```

16. Test your new nucleus.

    a. Use the IPL DDRXA utility with the DUMP NUC option to dump the nucleus on 123 to tape.

    b. Define your test XASRES volume as virtual 4A0. (If you use the sample HCPRIO in this book, you cannot use address 123 because that address is defined as a console.)

```
detach 0123 ■
define 423 04A0 ■
```

nnnn is the address of your test XASRES volume. The load deck should still be in your reader, since you have spooled it with the HOLD operand in a previous step.

    c. Set your virtual machine in XA mode and define your console at a valid virtual address in the HCPRIO file:

```
set machine xa ■
Ready; T=n.nn/n.nn hh:mm:ss
query cons ■
CONS vdevno ...
⋮
define vdevno 0020 ■
term conmode 3270 ■
```

    d. IPL your test XASRES volume *at second level.* Do *not* perform a hardware IPL.

```
ipl 4A0 clear ■
```

■ Press **ENTER** to create an interrupt.

If you can IPL your test XASRES volume successfully, continue with Step 6.

Note: You may need to define some temporary spool and paging space.

Step 6. Test the CP Nucleus

	Warning: You can do this step only if your test XASRES volume is a full-pack minidisk. If you are using a 10-cylinder minidisk, the only testing you can do is verifying that you can load the nucleus onto the test 423 minidisk. You did this when you IPLed your test XASRES in Step 5. Go to Step 7.	

	Now that you have applied service to all necessary parts, you should test the system to make sure that the problem has been fixed.	

1. Test your system at second level. If it performs to your satisfaction, do "Step 7. Put the New CP System into Production" on page 529. If the system fails, do not do Step 7. Find out what went wrong and start over with a new fix.

Step 7. Put the New CP System into Production

1. Shut down your second-level system. A Class A user (usually the primary system operator) must issue the SHUTDOWN command.

```
shutdown ■
SYSTEM SHUTDOWN STARTED
SYSTEM WARM START DATA SAVED
SYSTEM SHUTDOWN COMPLETE
HCPGIR450W CP ENTERED; DISABLED WAIT
            PSW 000A0000 00000961
```

2. Link to your production-level XASRES pack in read/write mode:

```
#cp link * 123 123 mr ■
Ready; T=n.nn/n.nn hh:mm:ss
```

3. Load (IPL) MAINT's virtual reader:

```
ipl 00c clear ■
```

```
hh:mm:ss * MSG FROM MAINT   :
  HCPGEN9010W NUCLEUS LOADED ON XASRES
  HCPGIR450W  CP ENTERED; DISABLED WAIT
  PSW 000A0000 00009010
```

The 9010 disabled wait state indicates that the CP nucleus is successfully loaded on the system residence device.

If you enter a disabled wait state with a code 691, you did not observe the warning notes above.

If you receive a different disabled wait state code, see *VM/XA SP System Messages and Codes Reference*.

Step 8. Build Utilities

	Do this step if:	
	• **You are processing program update service and have local modifications to utilities** • **You are processing corrective service** • **You are processing local service.**	

1. IPL 190:

```
ipl 190 clear ■
VM/XA CMS 5.5 mm/dd/yy hh:mm
■
```

2. Establish the appropriate minidisk access order:

```
setup ■
vmfsetup 56643082 cp (bld ■
```

3. Determine whether you have to build any utilities.
   - If you are processing corrective service, make and sort a list of the files you are servicing:

```
listfile hcp* text b (EXEC label ■
rename cms EXEC a hcptext = = ■
xedit hcptext EXEC ■
sort * 1 50 ■
file ■
```

The **EXEC** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

Check the whole list against Table 13 on page 531. You must use UTILITY to build any file you find listed there.

   - If you are processing program update service, use VMFVIEW to examine $VMFAPP $ERRLOG for files that have both IBM service and local modifications. Such files are identified by message DMSAPP1885W. Check these files against Table 13 on page 531. You must use UTILITY to build any file you find listed there.

4. For each file in the first column of Table 13 on page 531, type the line printed in the second column.

```
access 295 a ■
Ready; T=n.nn/n.nn hh:mm:ss
utility gen filename ■
Ready; T=n.nn/n.nn hh:mm:ss
```

The UTILITY EXEC builds the utilities on the A-disk (0295).

Table 13. UTILITY GEN Commands

Assemble File Name	Command to Type In	File Created by this Command
Note: If you receive service for a CP TEXT deck not listed in this table, it will be regenerated in some other way. You do not need to regenerate it now. (See Table 16 on page 595 in Appendix A, "VM/XA System Product Regeneration Requirements" for the procedure used. Text decks not listed in this table or in Table 16 on page 595 will become part of the nucleus when the system is built if they are listed in the CP load list.)		
HCPBSL	**utility gen 3card loader** ■	3CARD LOADER
HCPCCF, HCPFA*x*, HCPFON	**utility gen cpfmtxa** ■	CPFMTXA MODULE
HCPCCU	**utility gen hcpccu** ■	HCPCCU MODULE
HCPDDC, HCPDDR, HCPDDT, HCPDNC, HCPDNT	**utility gen ipl ddrxa** ■	IPL DDRXA
HCPDIR	**utility gen directxa** ■ **directxa user direct** ■	DIRECTXA MODULE
HCPED*x*	**utility gen dumpload** ■	DUMPLOAD MODULE
HCPIFC	**utility gen cperepxa** ■	CPEREPXA MODULE
HCPIMG	**utility gen genimage** ■	GENIMAGE MODULE
HCPLDL, HCPMDLAT	**utility cpload** ■	CPLOAD EXEC
HCPLDR	**utility gen hcpldr** ■	HCPLDR MODULE
HCPMOW	**utility gen monwrite** ■	MONWRITE MODULE
HCPNMT	**utility gen imagelib** ■	IMAGELIB MODULE
HCPOVE	**utility gen override** ■	OVERRIDE MODULE
HCPRET	**utility gen retrieve** ■	RETRIEVE MODULE
HCPSADMP	**utility gen hcpsadmp** ■	HCPSADMP MODULE

5. Test the utilities.
6. Copy the regenerated utilities from the A-disk to the CMS alternate BUILD1 (R-disk):

```
access 490 r ■
Ready; T=n.nn/n.nn hh:mm:ss

copy fn ft a = = r2 (olddate replace ■
Ready; T=n.nn/n.nn hh:mm:ss
```

ft can be MODULE, EXEC, LOADER, or DDRXA (see Table 13).

7. Erase the utilities from LOCAL1 (295) if they are not compatible with the utilities on the running S-disk. The Memo to Users will tell you whether or not they are compatible.
8. Repeat substeps 4 on page 530 through 7 for each utility.

9. IPL your system disk:

```
ipl 190 ■
VM/XA CMS 5.5 mm/dd/yy hh:mm
■
Ready; T=n.nn/n.nn hh:mm:ss
```

Step 9. Regenerate System Product Interpreter Programs

	Do this step if: • **You are processing program update service and have local modifications to System Product Interpreter programs (EXECs and XEDIT macros and the $DASD$ CONSTS file)** • **You are processing corrective service** • **You are processing local service.**	

1. Establish the appropriate minidisk access order:

   ```
   setup ■
   vmfsetup 56643082 cp (bld ■
   ```

2. Determine whether you have received service for any System Product Interpreter programs (EXECs or XEDIT macros and the $DASD$ CONSTS file):

   ```
   listfile * EXEC fm (EXEC ■
   rename cms EXEC a execfix = = ■
   listfile * XEDIT fm (EXEC ■
   rename cms EXEC a xeditfix = = ■
   listfile $DASD$ CONSTS fm (EXEC ■
   rename cms EXEC a dasdfix = = ■
   ```

   The updates are on the CP alternate LOCAL1 (295) disk if you are applying corrective service or on the CP alternate DELTA (594) disk if you are applying program update service.

   The **EXEC** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

3. Examine the list of EXECs and XEDIT macros to which you have applied service and determine if you have local modifications (including corrective service not included on the PUT you are processing) for any of them.

   Warning: The $VMFAPP $ERRLOG file will not identify these EXECs and XEDIT macros for you. You must know what local modifications you have made.

4. Copy the latest version of the base code for each EXEC or XEDIT macro for which you have both IBM service and local modifications to the CP alternate LOCAL1 disk (295).

   ```
   copy fn {$EXEC|$XEDIT|$CONSTS} fm = = b ■
   ```

5. Regenerate any EXECs or XEDIT macros for which you have received service and for which you have local modifications:

   ```
   execupdt fn ft b (ctl hcpxa options ■
   ```

   ft is EXEC, XEDIT, or CONSTS, *without the dollar sign*. HCPXA is the CP control filename.

   For any files with a SID code in columns 63 through 71, use the HIST and SID options. For all other files, use only the HIST option.

6. Erase the base code from the CP second alternate LOCAL1 disk:

   ```
   erase fn {$EXEC|$XEDIT|$CONSTS} b ■
   ```

7. Repeat substeps 4 through 6 for each System Product Interpreter program that needs regenerating.

8. For each EXEC or XEDIT macro for which you received service, whether you regenerated it or not, check to see if the original EXEC or XEDIT macro was on MAINT's 490 disk:

   ```
   listfile fn ft r ■
   ```

9. If the replaced or updated EXEC or XEDIT macro belongs on MAINT's 490 disk, copy it there:

   ```
   copy fn ft fm = = r2 (olddate replace ■
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   The macros are on the alternate LOCAL1 (295) disk if you are applying corrective service or on the alternate DELTA (594) disk if you are applying program update service.

Step 10. Test and Rebuild CMS

1. Perform Steps 10, 13, 15, 16, and 17 of Chapter 13, "Rebuilding CMS after Applying Service" on page 457 to make the files on the 490 disk available to users.

Step 11. Back Up the CP System

1. Use the procedure in Step 30 of Chapter 2, "Installing VM/XA System Product Release 2 with the Starter System (First Level)" to back up your new CP system.

	This is the end of the service procedure for CP.	

Chapter 15. Program Update Service or Corrective Service to the Dump Viewing Facility

This chapter describes:

- Receiving program update service or corrective service for the dump viewing facility
- Applying program update service or corrective service to the dump viewing facility
- Rebuilding the dump viewing facility.

Step 1. Preparation

1. Establish the appropriate minidisk access order: If you have not yet examined the SETUP EXEC and created your own version if necessary, turn to page 435 and do so now.

```
setup ■
Ready; T=n.nn/n.nn hh:mm:ss
```

2. Use the DASD DUMP Restore program to copy the current CMS BUILD1 disk to its corresponding alternate disk.

```
ddr ■
VM/XA SYSTEM PRODUCT DASD DUMP RESTORE PROGRAM
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS.
ENTER:
```

```
sysprint cons ■
ENTER:
```

This command tells DDR to send program messages to your console.

```
input 190 devtype MNT190 ■
ENTER:
```

190 is your current CMS system disk. *devtype* is the device type of the DASD volume where 190 is located.

```
output 490 devtype MNT490 ■
ENTER:
```

490 is your alternate CMS system disk. *devtype* is the device type of the DASD volume where 490 is located.

```
copy 000 endcyl ■
```

The value for *endcyl* depends on the device type of your 190 disk:

Device Type	*endcyl*
3350	73
3375	112
3380	71

```
DMKDDR711R Volid Read is MNT190.  Do you
           wish to continue?
yes ■
ENTER NEXT EXTENT OR NULL LINE:
■
⋮
END OF COPY
ENTER:
■
END OF JOB
```

You have not yet changed the label of the 490 minidisk. (You will do so in substep 3 on page 538.)

3. Relabel the 490 minidisk:

```
access 490 t ■
Ready; T=n.nn/n.nn hh:mm:ss
format 490 t (label ■
ENTER LABEL:
mnt490 ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Access 490 *after* the system disk. You *must* use the **label** option. If you don't, you will erase all the files on the 490 minidisk.

Step 2. Receive Service

Before You Receive Service

If you are not familiar with the new options provided in the VMFREC EXEC with the addition of APAR VM37518, refer to "VMFREC EXEC" on page 709. In particular, the section "Usage Notes" on page 710 describes considerations for using the TASK disk string to establish a separate disk string for the service EXECs.

1. Establish the appropriate minidisk access order:

   ```
   setup ■
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   ```
   release c ■
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   VMFREC INFO (MEMOS *xxx* loads documents to your C-disk if it is accessed, otherwise to your A-disk. You want them on your A-disk.

2. Attach a tape drive to MAINT:

   ```
   attach vdevno * 181 ■
   ```

3. Mount and ready the service tape.
4. Get the PUT or COR documentation and map the tape:

   ```
   vmfrec info (memos xxx ■
   DMSREC1851I This is n of n
                level level {PUT|COR} tape
   ```

   xxx is **put** if you are applying program update service, **cor** if you are applying corrective service. If the tape you mounted is not the kind of tape you specified, you receive an error message.

   VMFREC INFO (MEMOS maps the tape and loads the tape map, the PUT document or COR document, and the Memo to Users to your A-disk. Read these documents before going on.

   If your system default language is uppercase American English, you may get message DMSMGM814E. This message is caused by a problem with the REXX date function. You can ignore it.

5. VMFREC may tell you that the level of service EXECs on the tape is different than the level of service EXECs on your system. Refer to the "Usage Notes" section of the VMFREC EXEC description (on page 710) for information on receiving a new level of the service EXECs.
   - If you have not made any local modifications to the old product parameter file, you can use the new file as it stands. Go on to the next step.
   - If you had any local modifications in the old product parameter file, your override file should be on the CMS LOCAL1 disk string. Look at the new product parameter file on the 191 disk to see whether any changes from the last IBM-supplied product parameter file affect your override file. If

you are applying corrective service, be sure to include the changes from the corrective service override section of the base file in your override file.

The product parameter file has three sections for each component:

a. Control options
b. Minidisk assignments
c. EXECs needed to update each part of the product.

You may change the first two sections to suit your installation, but it is not recommended that you change the EXEC section.

Always place changes in an override file. Do not change the base product parameter file.

After you have made any necessary changes in your own override file, copy it to the 191 disk with the REPLACE option.

For more information about the product parameter file, see "The Product Parameter File" on page 401.

6. Receive the service files:

```
vmfrec 56643082 dv (xxx ■
```

xxx is **put** if you are applying program update service, **cor** if you are applying corrective service.

```
DMSREC1851I This is n of n
            level level {PUT|COR} tape
```

```
DMSREC1869R 56643082 DV begins on {PUT|COR}
             level volume vol.  Mount
             correct volume and press ENTER
             or type QUIT.
■
```

If you do not have the correct volume mounted, mount it now.

```
DMSREC1851I This is n of n
            level level {PUT|COR} tape
DMSREC1086I The current Service Diskmap contains
            the map of the mounted tape.
DMSREC1804I Receiving service for component DV of product
            56643082
DMSREC1851I Processing parttype with the part handler
            parthandler EXEC
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

7. Review the receive exception log ($VMFREC $ERRLOG). If necessary, correct any problems before going on.

```
vmfview receive ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Step 3. Apply the Updates

1. Establish the appropriate minidisk access order:

   setup ■

2. Apply the updates:

   vmfapply 56643082 dv (*xxx* ■
   DMSAPP1853I Processing PTF *ptfnum*
   DMSAPP1851I Processing *partname parttype* with
   the part handler *fn* EXEC
   ⋮
   Ready; T=*n.nn/n.nn hh:mm:ss*

   xxx is **put** if you are applying program update service, **cor** if you are applying corrective service.

3. Review the apply exception log ($VMFAPP $ERRLOG). If necessary, correct any problems before going on.

   vmfview apply ■

Step 4. Build the Dump Viewing Facility

1. Establish the appropriate minidisk access order:

   ```
   setup ■
   vmfsetup 56643082 dv (bld ■
   ```

2. Build the dump viewing facility:

   ```
   utility dvfgend all ■
   ```

3. Copy all MODULE files created by UTILITY DVFGEND ALL to the BUILD1 disk (490), then erase them from your A-disk.

   ```
   copy fn module a = = m2 ■
   Ready; T=n.nn/n.nn hh:mm:ss
   erase fn module a ■
   ```

4. Copy all MAP files created by UTILITY DVFGEND ALL to the APPLY disk (692) for program update service or the alternate LOCAL1 disk (395) for corrective service, then erase them from your A-disk.

   ```
   copy fn map a = = fm ■
   Ready; T=n.nn/n.nn hh:mm:ss
   erase fn map a ■
   ```

5. Copy DVF interface files to the CMS system disk:

   ```
   vmfbld 56643082 dv
      or
   vmfbld 56643082 cordv
   ```

   If you are processing PUT service, use **dv** as the component name.

   If you are processing COR service, use **cordv** as the component name.

   (To ensure that text decks are copied to the correct disk during VMFBLD execution using VMFBDCPY, **cordv** is now the component name for DV corrective service. This causes the corrective service override to be applied to the 56643082 $PPF and the DELTA disk string to be redefined).

Step 5. Regenerate System Product Interpreter Programs

	Do this step if:	
	• **You are processing program update service and have local modifications to System Product Interpreter programs (EXECs and XEDIT macros)** • **You are processing corrective service** • **You are processing local service.**	

1. Establish the appropriate minidisk access order:

   ```
   setup ■
   vmfsetup 56643082 dv (bld ■
   ```

2. Determine if you have received service for any System Product Interpreter programs (EXECs or XEDIT macros):

   ```
   listfile * EXEC fm (EXEC ■
   rename cms EXEC a execfix = = ■
   listfile * XEDIT fm (EXEC ■
   rename cms EXEC a xeditfix = = ■
   ```

   The updates are on the CMS alternate LOCAL1 (395) disk if you are applying corrective service or on the CMS alternate DELTA (593) disk if you are applying program update service.

   The **EXEC** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

3. Examine the list of EXECs and XEDIT macros to which you have applied service and determine if you have local modifications (including corrective service not included on the PUT you are processing) for any of them.

   Warning: The $VMFAPP $ERRLOG file does not identify these EXECs and XEDIT macros for you. You must know what local modifications you have made.

4. Copy the latest version of the base code for each EXEC or XEDIT macro for which you have both IBM service and local modifications to the CMS alternate LOCAL1 disk (395).

   ```
   copy fn {$EXEC|$XEDIT} fm = = b ■
   ```

5. Regenerate any EXECs or XEDIT macros for which you have received service and for which you have local modifications:

   ```
   execupdt fn ft b (ctl hcsxa options ■
   ```

   ft is EXEC or XEDIT, *without the dollar sign.* HCSXA is the dump viewing facility control file name.

   For any files with a SID code in columns 63 through 71, use the HIST and SID options. For all other files, use only the HIST option.

6. Erase the base code from the CMS alternate LOCAL1 disk:

   ```
   erase fn {$EXEC|$XEDIT} b ■
   ```

7. Repeat substeps 4 through 6 for each System Product Interpreter program that needs regenerating.

8. For each EXEC or XEDIT macro for which you received service, whether you regenerated it or not, check to see whether the original EXEC or XEDIT macro was on MAINT's 490 disk:

   listfile *fn ft* **m** ■

9. If the replaced or updated EXEC or XEDIT macro belongs on MAINT's 490 disk, copy it there:

   copy *fn ft fm* **= = m2 (olddate replace** ■
   Ready; T=*n.nn/n.nn hh:mm:ss*

   The macros are on the alternate LOCAL1 (395) disk if you are applying corrective service or on the alternate DELTA (593) disk if you are applying program update service.

Step 6. Test and Rebuild CMS

1. Perform Steps 10, 13, 15, 16, and 17 of Chapter 13, "Rebuilding CMS after Applying Service" on page 457 to make the files on the 490 disk available to users.

Step 7. Back Up the Dump Viewing Facility

1. Use the procedure in Step 30 of Chapter 2, "Installing VM/XA System Product Release 2 with the Starter System (First Level)" to back up the dump viewing facility.

	This is the end of the service procedure for the dump viewing facility.	

Chapter 16. Program Update Service or Corrective Service to GCS

This chapter describes:

- Receiving program update service or corrective service for the group control system
- Applying program update service or corrective service to the group control system
- Rebuilding the group control system.

Step 1. Preparation

1. Establish the appropriate minidisk access order. If you have not yet examined the SETUP EXEC and created your own version if necessary, turn to page 435 and do so now.

   setup ■
   Ready; T=*n.nn/n.nn hh:mm:ss*

2. Use the DASD DUMP Restore program to copy the current CMS BUILD1 disk to its corresponding alternate disk.

   ddr ■
   VM/XA SYSTEM PRODUCT DASD DUMP RESTORE PROGRAM
   ENTER CARD READER ADDRESS OR CONTROL STATEMENTS.
   ENTER:

   sysprint cons ■
   ENTER:

   This command tells DDR to send program messages to your console.

   input 190 *devtype* **MNT190** ■
   ENTER:

   190 is your current CMS system disk. *devtype* is the device type of the DASD volume where 190 is located.

   output 490 *devtype* **MNT490** ■
   ENTER:

   490 is your alternate CMS system disk. *devtype* is the device type of the DASD volume where 490 is located.

   copy 000 *endcyl* ■

   The value for *endcyl* depends on the device type of your 190 disk:

Device Type	*endcyl*
3350	73
3375	112
3380	71

```
DMKDDR711R Volid Read is MNT190.  Do you
           wish to continue?
yes ■
ENTER NEXT EXTENT OR NULL LINE:
■
⋮
END OF COPY
ENTER:
■
END OF JOB
```

You have not yet changed the label of the 490 minidisk. (You will do so in substep 3 on page 548.)

3. Relabel the 490 minidisk:

```
access 490 t ■
Ready; T=n.nn/n.nn hh:mm:ss
format 490 t (label ■
ENTER LABEL:
mnt490 ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Access 490 *after* the system disk. You *must* use the **label** option. If you don't, you will erase all the files on the 490 minidisk.

Step 2. Receive Service

Before You Receive Service

If you are not familiar with the new options provided in the VMFREC EXEC with the addition of APAR VM37518, refer to "VMFREC EXEC" on page 709. In particular, the section "Usage Notes" on page 710 describes considerations for using the TASK disk string to establish a separate disk string for the service EXECs.

1. Establish the appropriate minidisk access order:

   ```
   setup ■
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   ```
   release c ■
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   VMFREC INFO (MEMOS *xxx* loads documents to your C-disk if it is accessed, otherwise to your A-disk. You want them on your A-disk.

2. Attach a tape drive to MAINT:

   ```
   attach vdevno * 181 ■
   ```

3. Mount and ready the service tape.

4. Get the PUT or COR documentation and map the tape:

   ```
   vmfrec info (memos xxx ■
   DMSREC1852I This is n of n,
                  level level {PUT|COR} tape
   ```

   xxx is **put** if you are applying program update service, **cor** if you are applying corrective service. If the tape you mounted is not the kind of tape you specified, you receive an error message.

   VMFREC INFO (MEMOS maps the tape and loads the tape map, the PUT document or COR document, and the Memo to Users to your A-disk. Read these documents before going on.

   If your system default language is uppercase American English, you may get message DMSMGM814E. This message is caused by a problem with the REXX date function. You can ignore it.

5. VMFREC may tell you that the level of service EXECs on the tape is different from the level of service EXECs on your system. Refer to the "Usage Notes" section of the VMFREC EXEC description (on page 710) for information on receiving a new level of the service EXECs.

   - If you have not made any local modifications to the old product parameter file, you can use the new file as it stands. Go on to the next step.
   - If you had any local modifications in the old product parameter file, your override file should be on the CMS LOCAL1 disk string. Look at the new product parameter file on the 191 disk to see if any changes from the last IBM-supplied product parameter file affect your override file. If you are

applying corrective service, be sure to include the changes from the corrective service override section of the base file in your override file.

The product parameter file has three sections for each component:

a. Control options
b. Minidisk assignments
c. EXECs needed to update each part of the product.

You may change the first two sections to suit your installation, but it is not recommended that you change the EXEC section.

Always place changes in an override file. Do not change the base product parameter file.

After you have made any necessary changes in your own override file, copy it to the 191 disk with the REPLACE option.

For more information about the product parameter file, see "The Product Parameter File" on page 401.

6. For each GCS system to which you are applying service, create a product parameter override file using an alternate name for the system; for example, ALTGCS. Add the alternate name to the :OVERLST. tag. Specify the appropriate load list name after the :BLD. tag. Be sure to include any overrides from the :CORGCS. section of the base product parameter file.

7. Receive the service files:

```
vmfrec 56643082 gcs (xxx ■
```

xxx is **put** if you are applying program update service, **cor** if you are applying corrective service.

```
DMSREC1852I This is n of n,
              level level {PUT|COR} tape
DMSREC1806I The current Service Diskmap contains
              the map of the mounted tape

DMSREC1869R 56643082 GCS begins on {PUT|COR}
              level volume vol.  Mount
              correct volume and press ENTER or type QUIT.
■
DMSREC1852I This is n of n,
              level level {PUT|COR} tape
DMSREC1806I The current Service Diskmap contains
              the map of the mounted tape
DMSREC1804I Receiving service for component GCS
              of product 56643082
DMSREC1851I Processing parttype with the part handler
              parthandler EXEC
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

If you do not have the correct volume mounted, mount it now.

8. Review the receive exception log ($VMFREC $ERRLOG). If necessary, correct any problems before going on.

```
vmfview receive ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Step 3. Apply the Updates

1. Establish the appropriate minidisk access order:

   setup ■

2. Apply the updates:

   vmfapply 56643082 gcs (*xxx* ■
   DMSAPP1853I Processing PTF *ptfnum*
   DMSAPP1851I Processing *partname parttype* with
   the part handler *fn* EXEC
   ⋮
   Ready; T=*n.nn/n.nn hh:mm:ss*

   xxx is **put** if you are applying program update service, **cor** if you are applying corrective service.

3. Review the apply exception log ($VMFAPP $ERRLOG). If necessary, correct any problems before going on.

   vmfview apply ■

Step 4. Build a New GCS Macro Library

1. Establish the correct minidisk access order:

```
setup ■
vmfsetup 56643082 gcs (bld ■
```

Program Update Service Only

2. If you are processing program update service, erase GCSNEW MACLIB if it exists. **Do not erase GCSNEW MACLIB if you are processing corrective service or local service.**

```
erase gcsnew maclib fm ■
Ready; T=n.nn/n.nn hh:mm:ss
```

End of Program Update Service Only

	Do the rest of this step if: • **You are processing program update service and have local modifications to GCS macros or control blocks** • **You are processing corrective service** • **You are processing local service.**	

3. Access 495 as A:

```
access 495 a ■
```

4. Find out which macros and control blocks have been updated by IBM. Look for replacement macros and copy files on the service disk:

```
listfile * macro fm (EXEC ■
Ready; T=n.nn/n.nn hh:mm:ss
rename cms EXEC a gcsmcfix = = ■
Ready; T=n.nn/n.nn hh:mm:ss
listfile * copy fm (EXEC ■
Ready; T=n.nn/n.nn hh:mm:ss
rename cms EXEC a gcscyfix = = ■
Ready; T=n.nn/n.nn hh:mm:ss
```

The updates are on the alternate LOCAL1 (495) disk if you are applying corrective service or on the alternate DELTA (896) disk if you are applying program update service. You may also have updates on other disks in the LOCAL1 string.

The **EXEC** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

5. Copy the macros and control files that have been serviced to your A-disk:

```
copy fn macro fm = = a (replace ■
copy fn copy fm = = a (replace ■
```

6. Create or edit an EXEC to generate a macro library for updated macros called GCSNEW EXEC. It should have the same format as CSISP EXEC, and should list:
   - For program update service, only those macros and control blocks for which you have received service and for which you also have local modifications

- For corrective service, all macros and control blocks for which you have received service, plus any for which you have local modifications
- For local service, all macros and control blocks for which you have local modifications.

If you have no macros or control block files that must be listed in GCSNEW EXEC, go to substep 12 on page 554.

a. If you have not already created GCSNEW EXEC, issue:

```
copy csisp EXEC fm gcsnew = a (recfm f lrecl 80 ■
```

b. If you already have a copy of GCSNEW EXEC, issue:

```
copy gcsnew EXEC fm = = a (recfm f lrecl 80 ■
```

Now edit GCSNEW EXEC, adding the necessary macro and control block file names. If you created CMSNEW EXEC by copying CSISP EXEC, use the contents of CSISP EXEC only for a guide to the format of the macro entries. After you have added the macro and control block names you need to service, in the proper format, delete the macro names that came from CSISP EXEC.

```
xedit gcsnew EXEC a ■
⋮
file ■
```

7. You now need to copy every macro and control block file listed in GCSNEW EXEC to your A-disk.

```
copy fn macro fm = = a (olddate replace ■
copy fn copy fm = = a (olddate replace ■
```

8. Generate the new macro library by using the VMFMAC EXEC:

```
vmfmac gcsnew csixa ■
DMSUPD178I Updating macroname MACRO A1
DMSUPD178I Applying macroname update A1
macroname MACRO ADDED
⋮
Ready; T=n.nn/n.nn hh:mm:ss
```

Note: An alternate method to using the VMFMAC EXEC is to update the control blocks and macros by using the UPDATE module and issuing MACLIB DELETE and then MACLIB ADD commands for the MACLIB containing the changed macro or control block.

See *VM/XA SP CMS Command Reference* for details on the MACLIB command.

9. The new macro library is now on your A-disk (295). You can now erase GCSNEW COPY (a file created by VMFMAC) from the 295 disk.

```
erase gcsnew copy a ■
```

10. Add the new macro library name to the TEXT MACS card in the CSIXA CNTRL file. Make it the first macro library listed so that you use the updated macros.

```
xedit csixa cntrl ■
locate/TEXT MACS ■
TEXT MACS CSISP ...
change/MACS/MACS GCSNEW/ ■
TEXT MACS GCSNEW CSISP ...
file ■
```

11. If you want to make the new macro library available to general users, copy it to the GCS alternate BUILD1 disk (895):

```
copy gcsnew maclib a = = 12 (replace ■
```

12. Reaccess 495 as B and 191 as A:

```
access 495 b ■
access 191 a ■
```

Step 5. Update the GCS Message Repository

	Do this step if: • **You are processing program update service and have local modifications to the GCS message repository** • **You are processing corrective service and have local modifications to the GCS message repository** • **You are processing local service.**	

1. Determine your system default national language:

   ```
   query lang ■
   ```

   langid — See Table 5 on page 317 to identify the language corresponding to *langid*. Note the country code for that language.

2. Invoke the VMFNLS EXEC to update the GCS message repository:

   ```
   vmfnls csimesy repos 56643082 gcs ■
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   y is the country code for your system national language.

3. If you have updates to the message repositories for any other national languages installed on your system, repeat substep 2 for each language.

Step 6. Assemble and Build the GCS Nucleus

1. Establish the appropriate minidisk access order:

   setup ■
   vmfsetup 56643082 gcs (all ■

2. If you did not create the TESTNSS EXEC in "Step 9. Create Test EXECs and Files" on page 478 of Chapter 13, "Rebuilding CMS after Applying Service," do so now.

3. Define a new named saved system with an alternate name:

   testnss *altsysname* ■

   altsysname is an alternate name for your GCS system, for example, ALTGCS.

   ```
   HCPNSD440I The Named Saved System (NSS) altsysname
              was successfully defined
              in fileid fileno
   OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME   FILETYPE ORIGINID
   *NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMS        NSS      MAINT
   *NSS     nnnn NSS  A  nnnn mm/dd hh:mm:ss CMSXA      NSS      MAINT
   :
   *NSS     nnnn NSS  S  nnnn mm/dd hh:mm:ss altsysname NSS      MAINT
   ```

4. Create a configuration file for the GCS system you are servicing.

   copyfile gcs group b *altsysname* **= = (replace** ■
   xedit *altsysname* **GROUP** ■
   locate /SYSNAME= ■
   change /SYSNAME=*sysname***/SYSNAME=***altsysname***/** ■
   locate /SEGMENTNAME= ■
   change /SEGMENTNAME=*sysname***/SEGMENTNAME=***altsysname***/** ■
   file ■

5. Create an ASSEMBLE file for the GCS system you are servicing.

   copyfile *altsysname* **group b = assemble = (replace** ■

6. Use the VMFHASM EXEC to update and assemble the ASSEMBLE file.

   vmfhasm *altsysname* **56643082 gcs** ■

   ```
   DMSUPD181E No update files were found
   ASSEMBLING altsysname
   *** altsysname TEXT CREATED! ***
   PRT FILE fileno SENT FROM MAINT PRT AS fileno
      RECS nnnn COPY 001 A NOHOLD NOKEEP
   Ready; T=n.nn/n.nn hh:mm:ss
   ```

   These are informational messages. You can ignore them.

7. Copy the TEXT file from your A-disk to the alternate LOCAL1 disk:

   copy *altsysname* **text a = = b (replace** ■
   Ready; T=*n.nn/n.nn hh:mm:ss*

8. Issue the following commands:

```
change rdr class n class m ■
Ready; T=n.nn/n.nn hh:mm:ss
spool pun * class n nocont noeof ■
Ready; T=n.nn/n.nn hh:mm:ss
spool rdr class n keep ■
Ready; T=n.nn/n.nn hh:mm:ss
close pun ■
Ready; T=n.nn/n.nn hh:mm:ss
```

9. XEDIT the GCSLOAD EXEC so that it will pick up the text deck you created in substep 6 on page 556 for the system you are now servicing:

```
xedit gcsload EXEC ■
```

```
locate /&3 GCS ■
change /&3 GCS/&3 altsysname/ ■
file ■
```

If you have already edited GCSLOAD EXEC to service another GCS system, you must locate and change the *altsysname* for that system instead of GCS.

10. XEDIT the product parameter file to add *altsysname* to the :OVERLST. tag and to add an override section for *altsysname*:

```
xedit 56643082 $ppf ■
locate /:OVERLST. ■
change /CORGCS/CORGCS altsysname/ ■
bottom ■
input ■
```

```
:altsysname. GCS
```

This is a sample.

```
   or
```

Use GCS for PUT service, and CORGCS for COR service.

```
:altsysname. CORGCS
:BLD. REPLACE
 GCSLOAD VMFBDNUC BUILD1
:END. ■
```

(To ensure that text decks are copied to the correct disk during VMFBLD execution using VMFBDCPY, **CORGCS** is now the component name for GCS corrective service. This causes the corrective service override to be applied to the 56643082 $PPF and the DELTA disk string to be redefined).

```
■
file ■
```

Press **ENTER** to return to the command line.

11. Invoke VMFBLD to build the GCS nucleus.

```
vmfbld 56643082 altsysname (punch ■
```

```
0000001 FILE CHANGED
DMSBLD1851I Processing altsysname with
             the part handler VMFBDNUC EXEC
LOAD LIST: $$$TLL$$ EXEC A1 (MNT191)


RDR FILE fileno1 SENT FROM MAINT PUN AS fileno2
   RECS nnnn COPY 001 N NOHOLD NOKEEP
```

12. Review the build exception log ($VMFBLD $ERRLOG). If necessary, correct any problems before going on.

```
vmfview build ■
```

13. Issue the following commands:

```
spool pun off eof ■
Ready; T=n.nn/n.nn hh:mm:ss
close rdr ■
Ready; T=n.nn/n.nn hh:mm:ss
spool rdr class n keep ■
Ready; T=n.nn/n.nn hh:mm:ss
```

14. IPL your reader to save the new GCS system in the named saved system you created:

```
ipl 00c clear ■
MSG FROM MAINT: CSIIN134I altsysname has nnnnnn
  bytes of available common free storage
HCPNSD440I The Named Saved System (NSS) altsysname
            was successfully defined in fileid fileno


PRT FILE fileno1 SENT FROM MAINT RDR AS fileno2
   RECS nnnn COPY 001 N NOHOLD NOKEEP
System reset
```

This printer file is the GCS load map. Note the second file number.

15. IPL CMS (or ALTCMS):

```
ipl {cms|altcms} ■
VM/XA CMS 5.5 mm/dd/yy hh:mm
■
Ready; T=n.nn/n.nn hh:mm:ss
```

16. Save and print the GCS load map:

```
#cp transfer prt fileno2 to * rdr ■
access 595 e ■
receive fileno2 gcsnuc map e ■
```

Once you have the load map saved on disk, you can print a copy of it by issuing the following command:

```
print gcsnuc map e ■
```

17. Repeat substeps 3 on page 556 through 16 for each GCS system you are servicing.

Step 7. Test and Rebuild CMS

1. Perform Steps 10, 13, 15, 16, and 17 of Chapter 13, "Rebuilding CMS after Applying Service" on page 457 to make the files that VMFBLD EXEC loaded to the 490 disk in Step 6 available to users.

Step 8. Test and Rebuild the GCS Nucleus

Test the GCS nucleus you just built. When you are satisfied, repeat Step 6 to rebuild the nucleus with its original name.

Step 9. Back Up the GCS Named Saved Systems

1. Use the procedure in Step 29 of Chapter 2, "Installing VM/XA System Product Release 2 with the Starter System (First Level)" to back up the GCS named saved systems.

	This is the end of the service procedure for GCS.	

Chapter 17. Program Update Service to Licensed Programs

This chapter describes:

- Applying program update service to licensed programs that have a product service EXEC rather than a product parameter file.

To apply service to licensed programs, refer to the publications for the specific licensed program. If the licensed program has a product parameter file, the general concepts described in this book for program update service are applicable. To apply corrective service, see the publications for the specific licensed program.

If the licensed program has a product service EXEC, the following general procedure is applicable.

1. Establish the minidisk access order:

   setup ■

2. Mount and ready the program update tape.
3. Get the PUT documentation and map the tape:

   vmfrec info (memos ■

   VMFREC INFO (MEMOS maps the tape and loads the tape map, the PUT document or COR document, and the Memo to Users to your A-disk. Read these documents before going on.

4. Receive and apply service:

   vmfrec *prodid parameters* ■

   VMFREC invokes the product service EXEC to receive and apply service. You can pass parameters to the product service EXEC.

5. VMFREC may tell you that the level of service EXECs on the tape is different than the level of service EXECs on your system. Refer to the "Usage Notes" section of the VMFREC EXEC description (on page 710) for information on receiving a new level of the service EXECs.
   - If you have not made any local modifications to the old product parameter file, you can use the new file as it stands. Go on to the next step.
   - If you had any local modifications in the old product parameter file, your override file should be on the CMS LOCAL1 disk string. Look at the new product parameter file on the 191 disk to see whether any changes from the last IBM-supplied product parameter file affect your override file. If you are applying corrective service, be sure to include the changes from the corrective service override section of the base file in your override file.

     The product parameter file has three sections for each component:

     a. Control options
     b. Minidisk assignments
     c. EXECs needed to update each part of the product.

You may change the first two sections to suit your installation, but it is not recommended that you change the EXEC section.

Always place changes in an override file. Do not change the base product parameter file.

Warning: The disk addresses in the SPLOAD PROFILE, user directory, 56643082 $PPF, and SETUP EXEC, and the default addresses used by the ITASK EXEC must all match. If you change an address in any one of these places, you must change it in all the others. We recommend that you do not change any addresses.

After you have made any necessary changes in your own override file, copy it to the 191 disk with the REPLACE option.

For more information about the product parameter file, see "The Product Parameter File" on page 401.

6. Check the Program Directory for the licensed product (not the VM/XA SP Program Directory) or other licensed program documentation to see whether the product service EXEC, which VMFREC invoked for you in step 4 on page 563, builds the licensed program. For most products, it does. If it does not, invoke VMFBLD to build the licensed program:

vmfbld *prodid parameters* ■	VMFBLD invokes the product service EXEC with the BUILD parameter. The product service EXEC then rebuilds any part of the licensed program changed by service that requires a build function. You can pass parameters to the product service EXEC.

For compatibility, the VMSERV EXEC continues to be available to service this type of licensed program.

Chapter 18. Receiving and Applying Local Service

This chapter describes:

- Step-by-step procedures for applying local service to all components of VM/XA System Product using update files.

Introduction

Local service is defined as any service that is applied to your VM/XA SP system that was not supplied by IBM on a corrective service (COR) tape or program update (PUT) tape.

Warning: The application of local service can be a complicated and error-prone procedure because of the many variables that are involved. IBM strongly advises its customers to order service on a COR tape through the IBM Support Center whenever possible. When it is absolutely necessary to apply service from IBM before it is available on a COR tape, or a local modification is required to tailor a system environment, you must apply the service locally. This should be done very carefully.

Note: Before attempting to apply local service, please read Chapter 6, "VM/XA System Product Service—An Overview," Chapter 7, "How VM/XA System Product Uses Control Files and Update Files," and Chapter 8, "Files Used in Program Update Service and Corrective Service" of this book. It is very important to have a good understanding of how the service process works before you proceed with this chapter.

Local service can be placed in two categories:

- IBM local service includes any service that you receive from IBM that is not on either a COR or a PUT tape. Two common types of IBM local service are:
  - Emergency service from the change team

    When a severe problem arises and you cannot wait for a COR tape, you can get emergency service from the change team. This service can be sent to you on a tape or read to you over the phone. In either case, the service is **not** in the format required by the VMFREC and VMFAPPLY EXECs. To receive and apply this service, you **must** do it manually using the instructions in this chapter.
  - Service Received from INFO Access

    Any fixes obtained through INFO Access can also be obtained on a COR tape. If you choose to order a COR tape, it can be received and applied just like any other COR tape, with the automated service tools VMFREC and VMFAPPLY. If you choose not to order a COR tape, you **must** receive and apply it manually using the instructions in this chapter.
- Customer local service includes any service that is **not** supplied by IBM. Customer local service is used to tailor your VM/XA SP system. An example of customer local service is given in Step 16 of Chapter 2, "Installing VM/XA System Product Release 2 with the Starter System (First Level)." The example given shows how to tailor the HCPBOX ASSEMBLE file. To receive and apply this service, you **must** do it manually using the instructions in this chapter.

The instructions in this chapter apply to all components of the VM/XA SP system (CMS, CP, DV, and GCS). Whenever there are variations for different components, they are noted.

Step 1. Preparation (For CMS Local Service Only)

In order to be consistent with the philosophy of not building on top of your production system, you must perform the following steps before applying local service to CMS:

1. Copy the latest version of the DMSNGP ASSEMBLE file to the 395 disk. (If there is already a copy of DMSNGP ASSEMBLE or DMSNGP TEXT on the 395 disk, rename it first.) Edit the copy and make the following changes:

   a. Change SYSDISK = 190 to SYSDISK = 490.

   b. Change IPLADDR = 190 to IPLADDR = 490.

   c. Change HELP = 19D to HELP = 49D (or HELP = 19C to HELP = 49C).

   d. Change SYSNAME = CMS to SYSNAME = *newname*, where *newname* is the name you choose for your test system. The sample procedure in Chapter 13, "Rebuilding CMS after Applying Service," uses ALTCMS.

   e. Change INSTSEG = CMSINST to INSTSEG = *newname*, where *newname* is the name you choose for your test installation segment. The sample procedure in Chapter 13, "Rebuilding CMS after Applying Service," uses ALTINST.

   f. Change the VERSION = and INSTID = parameters to identify your test system.

   g. Make sure that SAVESYS = NO. If SAVESYS = YES, when you IPL CMS the system will try to save the alternate CMS segment that you define in Step 10 of Chapter 13, "Rebuilding CMS after Applying Service" in the existing CMS segment.

   These changes cause the new CMS nucleus (with service) to be built on the 490 minidisk.

2. Use the DASD DUMP Restore program to copy the current BUILD disk to its corresponding alternate disk.

```
ddr ■
VM/XA SYSTEM PRODUCT DASD DUMP RESTORE PROGRAM
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS.
ENTER:
```

```
sysprint cons ■
ENTER:
```

This command tells DDR to send program messages to your console.

```
input 190 devtype MNT190 ■
ENTER:
```

190 is your current CMS system disk. *devtype* is the device type of the DASD volume where 190 is located.

```
output 490 devtype MNT490 ■
ENTER:
```

490 is your new CMS system disk. *devtype* is the device type of the DASD volume where 490 is located.

```
copy 000 endcyl ■
```

The value for *endcyl* depends on the device type of your 190 disk:

Device Type	*endcyl*
3350	73
3375	112
3380	71

```
DMKDDR711R Volid Read is MNT190.  Do you
           wish to continue?
yes ■
ENTER NEXT EXTENT OR NULL LINE:
■
⋮
END OF COPY
ENTER:
■
END OF JOB
```

You have not yet changed the label of the 490 minidisk. (You will do so in substep 3.)

3. Relabel the 490 minidisk:

```
access 490 t ■
Ready; T=n.nn/n.nn hh:mm:ss
format 490 t (label ■
ENTER LABEL:
mnt490 ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Access 490 *after* the system disk. You *must* use the **label** option. If you don't, you will erase all the files on the 490 minidisk.

4. Repeat substeps 2 on page 566 and 3 to copy the current HELP disks (19D, 19C, 19B...) to their corresponding alternate disks (49D, etc.) and to relabel the alternate disks as MNT49*n*. Check the minidisk definitions in your user directory to find the appropriate value for *endcyl*.

Step 2. Receive Service

1. Log on to MAINT and establish the appropriate minidisk access order:

```
setup ■
Ready; T=n.nn/n.nn hh:mm:ss
vmfsetup 56643082 compid (all ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Substitute for *compid* the component you are servicing (CMS, CP, DV, or GCS).

2. Merge the LOCAL1 minidisk string by copying the entire contents of the alternate LOCAL1 minidisk to the intermediate alternate LOCAL1 minidisk and erasing the alternate LOCAL1 minidisk.

   Note: For the addresses of the LOCAL1 minidisks for the component you are servicing, see Figure 6 on page 364.

```
copyfile * * b = = e (olddate replace ■
Ready; T=n.nn/n.nn hh:mm:ss
erase fn ft b ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Substitute for *fn* and *ft* the file name and file type of each file you have on the alternate LOCAL1. You may also use the ACCESS command with the ERASE option. If you choose to use ACCESS, after you issue the ACCESS command you must write a file to the disk in order to rewrite the file directory.

3. Access the alternate LOCAL1 as A in R/W mode and the intermediate alternate LOCAL1 as B/A (an extension of A) in read-only mode:

```
access vaddr1 a ■
DMSACC724I vaddr1 replaces A (191)
Ready; T=n.nn/n.nn hh:mm:ss
release e ■
Ready; T=n.nn/n.nn hh:mm:ss
access vaddr2 b/a ■
DMSACP723I B (vaddr2) R/O
Ready; T=n.nn/n.nn hh:mm:ss
```

Substitute for *vaddr1* the virtual address of the alternate LOCAL1 (295, 395, or 495) and for *vaddr2* the virtual address of the intermediate alternate LOCAL1 (591, 691, or 791).

4. Place the service that you receive in one or more CMS files on the alternate LOCAL1 minidisk.

Service Supplied on Magnetic Tape

a. If service is supplied on a magnetic tape, you must mount the tape, attach the tape to MAINT's virtual machine, and load the tape file onto the alternate LOCAL1 minidisk.

```
attach rdevno * 181 ■
TAPE rdevno ATTACHED TO MAINT 0181
Ready; T=n.nn/n.nn hh:mm:ss
```

rdevno is the real device number of the tape drive.

The exact wording of this response may vary, depending on who is actually mounting the tape for you.

```
vmfplc2 load (eot disk ■
LOADING...
END-OF-FILE OR END-OF-TAPE
Ready; T=n.nn/n.nn hh:mm:ss
```

The tape file is in VMFPLC2 format. The file created by this command is called TAPE MAP A, and lists the file names and file types of all files loaded to the alternate LOCAL1 minidisk.

```
rename tape map a lcl = = ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Print or rename TAPE MAP A. You need to refer to these files throughout the rest of the service procedure. Renaming the file prevents VMFPLC2 from overwriting it.

End of Service Supplied on Magnetic Tape

Service Supplied as Hard Copy or by Telephone

b. If service is supplied as hard copy or by telephone as the result of an emergency, your IBM Service Representative should supply you with the file name and file type of each required file. Make a list of these names. You must now type in these files. Use the XEDIT command to create each CMS file on the alternate LOCAL1 minidisk.

End of Service Supplied as Hard Copy or by Telephone

Step 3. Apply Service

Changes to the system are provided as new parts, replacement parts, or updates to existing parts initially shipped on the product tape. The following table shows the base file types that might be serviced and whether they are serviced by update or replacement service:

Table 14. Types of Files that Can Be Serviced

File type	Corresponding File	Type of Service
TEXT	With corresponding ASSEMBLE	Update
	Without corresponding ASSEMBLE	Replacement
EXEC	With corresponding $EXEC	Update
	Without corresponding $EXEC	Replacement
XEDIT	With corresponding $XEDIT	Update
	Without corresponding $XEDIT	Replacement
MACRO		Update
COPY		Update
Any other types		Replacement

For the following examples, assume that you have received emergency service from the change team. You have received one PTF, but it is not COR closed. A PTF is considered COR closed when it is made available for distribution on a COR tape. The PTF (UM34567) hits two modules, HCPABC and HCPXYZ. Also assume that you are creating customer local service for HCPQRS. These examples are for CP files, but the procedures for all components are the same.

Note: Since GCS and DV are object-maintained, you do not have to worry about update service for these components.

For the list of files that you are servicing, refer to the tape map (LCL MAP A) from the last step if you received service from tape, or to the list that your IBM Service Representative gave you if you received service as hard copy or over the phone.

1. Determine the base file types of HCPABC, HCPXYZ and HCPQRS.

   For each unique file name that you are servicing, issue the LISTFILE command. Using Table 14 and "VM/XA SP File Types and Abbreviations" on page 861 as a guide, write down what the base file type is for each file name being serviced.

```
listfile hcpabc * * ■
HCPABC   H98765HP  A1
HCPABC   AUXLCL    fm
HCPABC   L00001LC  fm
HCPABC   LCL00001  fm
HCPABC   AUXXA     fm
HCPABC   H12345HP  fm
HCPABC   TXT32123  fm
HCPABC   ASSEMBLE  fm
HCPABC   TEXT      fm
Ready; T=n.nn/n.nn hh:mm:ss
```

   In this example, you can see both an ASSEMBLE file and a corresponding TEXT file. This indicates that HCPABC is serviced by updates (source-maintained).

```
listfile hcpxyz * * ■
HCPXYZ   TXT34567  A1
HCPXYZ   AUXXA     fm
HCPXYZ   H12345PP  fm
HCPXYZ   TXT32123  fm
HCPXYZ   TEXT      fm
Ready; T=n.nn/n.nn hh:mm:ss
```

In this example, there is no corresponding ASSEMBLE file, so the base file type is TEXT. This indicates that HCPXYZ is serviced by replacement (object-maintained). Do not be confused by the update file HCPXYC H12345PP. It is only an update shell. Update shells are explained below.

```
listfile hcpqrs * * ■
HCPQRS   EXEC      fm
HCPQRS   $EXEC     fm
Ready; T=n.nn/n.nn hh:mm:ss
```

In this example, there is an EXEC and a corresponding $EXEC. This indicates that HCPQRS is source update serviced.

2. Use XEDIT to create an AUX file entry on the AUXLCL level.

   There are three fields that make up an AUX entry. The first field contains the file type of the update file. Imbedded in this file type is the 5-digit numeric portion of the APAR number. When you originate an update, this 5-digit number should be a local tracking number. The second field is the level of the fix. For local service, use LCL in this field. If you want to keep your IBM local service separate from your customer local service, you can use LC*x* in this field (*x* is any alphanumeric character). The third field is the PTF number or a local tracking number.

   a. If an AUXLCL file already exists, as in the case of HCPABC, copy it to the alternate LOCAL1, which is currently accessed as A.

```
copyfile hcpabc auxlcl fm = = a ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Substitute for *fm* the file mode of the minidisk on which the AUXLCL file currently resides.

The existing HCPABC AUXLCL file has only one entry:

```
L00001LC LCL LC00001 * CUSTOMER LOCAL SERVICE TO CIRCUMVENT A PROBLEM
```

This is a locally originated fix. 00001 is your local tracking number.

If the new fix from IBM fixes the problem you circumvented with the entry in the AUXLCL file, replace the entry in the AUXLCL file with the new one from IBM. Your new AUX entry in the AUXLCL file is then:

```
H98765HP LCL UM34567 * IBM UPDATE TO FIX THE PROBLEM
```

If the new fix from IBM is unrelated to the entry that is in the AUXLCL file, you **must** review your previous customer local service and rework it if necessary. Put the AUX entry for the new fix below the existing AUXLCL entry in the file. Then use XEDIT with the CTL option to rework the old AUX entry. The new AUXLCL file would look like this:

```
L00001LC LCL LC00001 * REWORKED CUSTOMER LOCAL SERVICE
H98765HP LCL UM34567 * IBM UPDATE TO FIX A DIFFERENT PROBLEM
```

   b. If an AUXLCL file does not exist but another AUX file does, as in the case of HCPXYZ, you can copy and rename an existing AUX file to the alternate LOCAL1 which is currently accessed as 'A', and use it as a guide.

```
copyfile hcpxyz auxxa fm = auxlcl a ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Substitute for *fm* the file mode of the minidisk that the AUXXA file currently resides on.

For HCPXYZ, you have a replacement text deck. Use XEDIT to look at the self-documenting information at the top of the new text deck:

xedit hcpxyz txt34567 a

The system displays a file like the one in Figure 34.

```
H12345PP 201 UM32123 * THIS IS AN OLD FIX
*       HCPXYZ H12345PP fm nnnnnn mm/dd/yy hh:mm:ss
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
H98765PP xxx UM34567 * THIS IS THE NEW FIX
*       HCPXYZ H98765PP fm nnnnnn mm/dd/yy hh:mm:ss
* PREREQ: VM12345
* CO-REQ: NONE
* IF-REQ: NONE
:
```

Figure 34. A Sample Replacement Text Deck (HCPXYZ TXT34567 A)

In AUX files, the most recent entry is on top. In text decks, the most recent entry is on the bottom of the prolog. The first entry in the text deck is already in the HCPXYZ AUXXA file. The second entry is the new fix and requires that an AUX entry be made on the AUXLCL level. The AUX entry can be copied directly from the replacement text deck. The only change should be in the second field. As mentioned above, this field should be LCL, so the AUX entry in HCPXYZ AUXLCL should be:

```
H98765PP LCL UM34567 * THIS IS THE NEW FIX
```

c. If no AUX files exist, as in the case of HCPQRS, use XEDIT to create one on the alternate LOCAL1, which is currently accessed as A.

For HCPQRS, there are no existing updates, so when you have finished editing the AUXLCL file, the only entry in it should be the update that you are creating:

```
L00002LC LCL LC00002 * LOCAL FIX TO CIRCUMVENT A PROBLEM
```

Note: This process would be the same for other parts, including files with the file types ASSEMBLE, $EXEC, $XEDIT, MACRO, and COPY.

3. Create an update file or update shell.

Note: When VMFAPPLY creates an AUX entry for an update, the file mode of the update file is changed to 5. For local service, it is important that you do **not** change the file mode of the update files to 5. If VMFAPPLY tries to apply this same fix as part of a future corrective service application you get VMFAPPLY warning messages in the VMFAPPLY exception log. The result is that AUXCOR entries are not created for each text deck that has updates with file mode 5. If you want to use the VMFAPPLY convention of changing the update file mode to 5 so as to keep track of updates for which you have created AUX entries, you must keep manually applied local service and corrective service on separate disk strings (disk strings are defined in 56643082 $PPF). If you do this, subsequent corrective service is not affected. IBM recommends that its customers **not** do this.

a. For HCPABC, IBM shipped you the update file for the fix.

b. For HCPXYZ, IBM shipped you only a replacement text deck named HCPXYZ TXT34567. For this part, you must create an update shell. An update shell takes the place of the source update file that would exist if the part were on update service. Just as with the AUXLCL file for HCPXYZ, you can take the information for the update shell from the replacement text deck (see Figure 34). Your update shell for HCPXYZ should look like Figure 35 on page 573.

```
./ * PREREQ: VM12345
./ * CO-REQ: NONE
./ * IF-REQ: NONE
./ * DEPEND: NONE
./ * THIS IS AN UPDATE SHELL
```

Figure 35. A Sample Update Shell (HCPXYZ H98765PP A)

c. For HCPQRS, you must create your own update file because it is originated locally. Normally, you would already know what kind of change you wanted to make to HCPQRS. To create the update file, you will invoke XEDIT with the CTL option.

```
xedit hcpqrs $exec a (ctl hcpxa ■
:
file
Ready; T=n.nn/n.nn hh:mm:ss
```

If you receive message DMSXUP178I, previous updates are being applied before your new update is created. You should receive message DMSXUP180W. This message tells you that your new update cannot be found. This is because you are in the process of creating it. At this point you can make your updates. Remember that these updates do **not** go into the $EXEC file, but instead are put in the HCPQRS L00002LC update file.

Note: This process is the same for other parts, including files with the file types ASSEMBLE, $EXEC, $XEDIT, MACRO, and COPY. The CNTRL file varies from component to component. See Chapter 7, "How VM/XA System Product Uses Control Files and Update Files" on page 377 for a description of the CNTRL files and to find out which one is appropriate for the service you are performing.

4. Create executable parts.

   At this point, you should have a correct AUXLCL file and a corresponding update file or shell.

   a. For HCPABC, you still need a text deck. This text deck is generated when you assemble HCPABC during the rebuilding process after you have finished this chapter.

   b. For HCPXYZ, IBM supplied you with a replacement text deck. The text deck, however, is not named consistently with the AUXLCL entry that you created. You must now rename this text deck.

```
rename hcpxyz txt34567 a = lcl34567 = ■
Ready; T=n.nn/n.nn hh:mm:ss
```

   c. For HCPQRS, you still need an EXEC. This EXEC is generated when you run EXECUPDT for HCPQRS during the rebuilding process, after you have finished this chapter.

   The following lists the base file types that may be included in your local service, along with a brief description of how the executable part is generated.

   - ASSEMBLE with a corresponding TEXT or TXT*nnnnn*: Update

     You reassemble this part during the rebuilding process.

   - TEXT or TXT*nnnnn* with no corresponding ASSEMBLE: Replacement

     You must rename the new TXT*nnnnn* file to LCL*nnnnn*, where *nnnnn* is the PTF number.

```
rename file name txtnnnnn a = lclnnnnn = ■
Ready; T=n.nn/n.nn hh:mm:ss
```

- EXEC with a corresponding $EXEC: Update

  You will use EXECUPDT to generate the EXEC during the re-building process.

- EXEC with no corresponding $EXEC: Replacement

  If you did not receive a new file with the file type EXEC with your service, and you did receive a file with the file type of EXC*nnnnn*, where *nnnnn* is the PTF number, this file must be copied with the file type of EXEC:

```
copyfile file name excnnnnn a = exec = (olddate ■
Ready; T=n.nn/n.nn hh:mm:ss
```

- XEDIT with a corresponding $XEDIT: Update

  You will use EXECUPDT to generate the XEDIT macro during the rebuilding process.

- XEDIT with no corresponding $XEDIT: Replacement

  If you did not receive a new file with the file type XEDIT with your service, and you did receive a file with the file type of XED*nnnnn*, where *nnnnn* is the PTF number, this file must be copied with the file type of XEDIT:

```
copyfile file name xednnnnn a = xedit = (olddate ■
Ready; T=n.nn/n.nn hh:mm:ss
```

- MACRO: Update

  You will use VMFMAC to generate a MACLIB with the new MACRO during the rebuilding process.

- COPY: Update

  You will use VMFMAC to generate a MACLIB with the new COPY file during the rebuilding process.

- Any other types: Replacement

  If the new replacement parts do not already have their appropriate file type, then they have an abbreviated form of the file type concatenated with the PTF number. Use "VM/XA SP File Types and Abbreviations" on page 861 to find out the complete file type and copy the file with the complete file type:

```
copyfile file name xxxnnnnn a = ffffffff = (olddate ■
Ready; T=n.nn/n.nn hh:mm:ss
```

  xxx is the abbreviated file type, *nnnnn* is the PTF number, and *ffffffff* is the complete file type.

Note: After you rebuild the component to which you are applying local service, you may wish to do a manual merge of the LOCAL1 disk string. (For instructions, see Appendix G, "Controlling Disk String Merges" on page 859.) Because you have not run VMFREC to receive local service, no restart indicator files have been produced. If the next service you receive is from a corrective service tape, VMFREC does not perform the usual merge, so if you want it to be done you must do it yourself. If the next service you receive is from a program update tape, VMFREC merges normally.

MAP 0015: What to Do Next

001

Do you have a service tape?
Yes **No**

002

– Go to the instructions for rebuilding the component to which you applied local service:

CMS Chapter 13, "Rebuilding CMS after Applying Service" on page 457
CP Chapter 14, "Rebuilding CP after Applying Service" on page 515
DV Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility," "Step 4. Build the Dump Viewing Facility" on page 542
GCS Chapter 16, "Program Update Service or Corrective Service to GCS," "Step 4. Build a New GCS Macro Library" on page 552.

003

– Go to the instructions for receiving service for the component to which you applied local service:

CMS Chapter 9, "Receiving Program Update Service or Corrective Service for CMS" on page 435
CP Chapter 10, "Receiving Program Update Service or Corrective Service for CP" on page 443
DV Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility," "Step 2. Receive Service" on page 539
GCS Chapter 16, "Program Update Service or Corrective Service to GCS," "Step 2. Receive Service" on page 549.

Chapter 19. Emergency Local Service Using the Patch Facility

This chapter describes:

- Step-by-step procedures for applying patches to the CMS and CP nuclei.

Introduction

Before attempting to use the patch facility, please read the following sections of this book:

- Chapter 6, "VM/XA System Product Service—An Overview" on page 355
- Chapter 7, "How VM/XA System Product Uses Control Files and Update Files" on page 377
- Chapter 8, "Files Used in Program Update Service and Corrective Service" on page 385
- "HCPLDR Command" on page 613 in Appendix B, "EXEC and Command Format Summaries"
- "The Patch Facility" on page 640 in Appendix B, "EXEC and Command Format Summaries."

The patch facility is used to circumvent problems in text decks on replacement service (object-maintained) that are part of the CMS or CP nucleus. If the problem you have is in a file that is not in the nucleus, you cannot use the patch facility.

Patches to object code (TEXT files) exploit the capability of the HCPLDR module when VER, REP, and ICS statements are included in the replacement files.

The instructions in this chapter apply to CMS and CP. Whenever there are variations for different components they are noted.

Your IBM service representative will give you the specific hexadecimal codes that need to be changed.

Step 1. Receive the Patch

In the example below assume that you have identified the object-maintained part that is causing a problem in your VM/XA SP system. You call your IBM service representative, who tells you that there is no existing fix for the problem. He tells you that, in order to circumvent the problem, you can change the object code in HCPAFF at displacement X'061C' from X'0780' to X'0700'. This example is representative of the type of fix you may get. It changes a branch to a no-op. Finally, your representative tells you that he has assigned APAR number VM23877 to this problem. If your representative does not give you an APAR number, use a local tracking number.

1. Log on to MAINT and establish the appropriate minidisk accesses:

```
setup ■
Ready; T=n.nn/n.nn hh:mm:ss
vmfsetup 56643082 compid (all ■
Ready; T=n.nn/n.nn hh:mm:ss
```

Substitute for *compid* the component you are servicing (CMS or CP).

2. Create a patch update file on the alternate LOCAL1 minidisk:

```
xedit hcpaff p23877 b ■
input ■
:
file ■
```

Your patch update file may look like Figure 36.

Note that the spaces shown in the example are required spaces. For information on patch update file control statement syntax, see the section "The Patch Facility" on page 640.

Note that the cards appear as comments in the patch update file. The $VMFPAT$ EXEC translates the control cards into the format required by the HCPLDR module before it calls HCPLDR.

```
./ * * PREREQ: NONE
./ * * CO-REQ: NONE
./ * NAME HCPAFF
./ * * This is a patch.  It
./ * *   will be replaced by
./ * *   APAR VM23877.
./ * VER 61C 0780
./ * REP 61C 0700 ■
```

Figure 36. A Sample Patch Update File

Step 2. Apply the Patch

1. Use XEDIT to create an AUX file entry on the AUXPAT level.

   There are three fields that make up an AUX entry. The first field contains the file type of the update file. Imbedded in this file type is the 5-digit numeric portion of the APAR number of the fix. When you originate an update, this 5-digit number should be a local tracking number. The second field is the level of the fix. For patch service you **must** use TX$ in this field. Since there is no PTF number for the patch, the third field is the APAR number or a local tracking number.

   ```
   xedit hcpaff auxpat b ■
   input ■
   :
   file ■
   ```

   Your patch AUX file might look like Figure 36 on page 578.

```
P23877 TX$ VM23877 * A PATCH TO BE REPLACED BY APAR VM23877
```

Figure 37. A Sample Patch AUX File

Note: You would not normally have any service on the AUXLCL level for the part for which you are creating a patch, because service on the AUXLCL level is accomplished with source update files. If you have the source to update, a patch is not necessary. If, however, you do have service at the AUXLCL level, you must rework it so that it does not conflict with the patch you are applying. You must also rework any service you have on the AUXPAT level so that it does not conflict with the patch you are applying.

2. Check the control file you are using to make sure that it includes an AUX file identification record for the AUXPAT level of service.

   "Main Control Files" on page 380 shows the contents of the default CNTRL files. The default CNTRL files for the VM/XA System Product system include the AUXPAT level at the highest level of service. If you modify the CNTRL file structure, you must make sure that the AUX file that you create for this patch is in your modified structure.

MAP 0016: What to Do Next

001

– Go to the instructions for rebuilding the component to which you applied the patch:

CMS Chapter 13, "Rebuilding CMS after Applying Service" on page 457
CP Chapter 14, "Rebuilding CP after Applying Service" on page 515
DV Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility," "Step 4. Build the Dump Viewing Facility" on page 542
GCS Chapter 16, "Program Update Service or Corrective Service to GCS," "Step 4. Build a New GCS Macro Library" on page 552.

Chapter 20. Removing Service from VM/XA SP

This chapter describes:

- How to remove program update service, corrective service, or local service from any component of VM/XA SP.

If you have a problem with a PTF, or if a PTF arrives for which you have applied a local circumvention, you will need to remove service you have applied.

When you install program update service, corrective service, or local service to VM/XA System Product, you build your new system on different minidisks from your old system. You should still have your old system available so you can return to it simply by accessing the old disks. In this way, of course, you remove an entire level of service. If you wish to delete only certain PTFs from VM/XA System Product, follow the procedure in this chapter.

First you must understand that many parts of a component, possibly of different kinds, may be serviced by a single PTF. You can tell what kind of part an update affects by the part's file type:

File type	**Part**
ASSEMBLE	ASSEMBLE file
TEXT or TXT*nnnnn*	Text deck ("TEXT file")
EXEC or EXC*nnnnn*	EXEC
MACRO or MAC*nnnnn*	Macro
COPY or CPY*nnnnn*	Control block ("COPY file")
XEDIT or XED*nnnnn*	XEDIT macro
nnnnn is the number of the most recently applied PTF.	

The examples in this chapter are written to assist you in understanding what has to be done to remove service from VM/XA SP. They are not meant to cover all possible situations.

1. Make a list of all the PTFs you want to remove, leaving space to write in the corresponding updates. If you know the number of the APAR you want to remove, but not the corresponding PTF number, see Appendix I, "How To Find the PTF Number From the APAR Number" on page 865.

   In this example, we suppose that you want to remove only one PTF, UM00001, from CP.

2. With every PTF, you receive a parts list, which identifies all the parts affected by that PTF. Examine the parts list:

```
vmfsetup 56643082 cp (all ■
Ready; T=n.nn/n.nn hh:mm:ss
```

xedit um00001 $ptfpart *fm* ■

The file name of the parts list is the PTF number; the file mode is $PTFPART. In this example, UM00001 is applied from a corrective service tape and is therefore on the CP alternate LOCAL1 disk. If it had been applied from a program update tape, it would be on the CP alternate DELTA disk.

```
HCPABC H34567HP
HCPXYZ H34567HP
HCPABC TXT00001
```

UM00001 $PTFPART lists one update for HCPABC and one update for HCPXYZ, along with the HCPABC text deck for this PTF.

3. Since a text deck is shipped for HCPABC, it must be an ASSEMBLE file. To find out what kind of part HCPXYZ is, check its file type:

listfile hcpxyz * * ■

```
HCPXYZ   AUXCOR   fm
HCPXYZ   MACRO    fm
HCPXYZ   MAC00004 fm
HCPXYZ   MAC00001 fm
```

HCPXYZ is a macro. (Replacement macro libraries are not shipped on IBM corrective service tapes. Macros are pulled together into a macro library when service is applied with the VMFMAC command.)

4. Comment out or delete the update entries in the auxiliary control file at the service level affected by the PTF. (It is better to comment them out, so that you can see what you did.) If the update you are commenting out or deleting is **not** at the top of the auxiliary control file, then you must also comment out or delete any updates above the update you are removing. You must also add them to your list of updates to be removed if they are not already there.

xedit hcpabc auxcor ■

Since PTF UM00001 was applied as corrective service, it should be listed in the auxiliary control file for the corrective service level. If you are using the main control file supplied by IBM, the file type of that auxiliary control file is AUXCOR.

```
*H88888HP COR UM00003
*H56789HP COR UM00002
*H34567HP COR UM00001
 H12345HP COR UM00000
```

The update for PTF UM00001 is listed on the third line from the top.

To comment out an entry, put an asterisk in front of it. Remember to comment out the entries **above** the one you want to remove too.

file ■

xedit hcpxyz auxcor ■

```
*H99999HP COR UM00004
*H34567HP COR UM00001
```

file ■

5. When you edited HCPABC AUXCOR and HCPXYZ AUXCOR, you found three more updates that must be removed. Add these updates, along with the corresponding PTF number in the third word of the AUX entry, to your list of fixes to delete. Your list of updates that must be deleted now contains:

```
HCPABC H34567HP   PTF UM00001
HCPXYZ H34567HP   PTF UM00001
HCPABC H88888HP   PTF UM00003
HCPABC H56789HP   PTF UM00002
HCPXYZ H99999HP   PTF UM00004
```

   In addition to removing the updates on this list, you must also review the parts lists for the corresponding PTFs (UM00003, UM00002, and UM00004) to find all the additional updates in those PTFs. You must remove the whole PTF, not just part of it. Save that task for later. For now, just note that you will have to look at these additional PTFs after you finish removing the one you set out to remove.

6. If the PTF you are removing does not affect any text decks, go to Step 7 on page 584. If it does, as in this example, determine whether the text deck for which you are removing service is in the load list for this component (CPLOAD EXEC, CMSLOAD EXEC, or GCSLOAD EXEC):

```
xedit cpload EXEC ■
locate /hcpabc ■
qquit ■
```

   - If the text deck for which you are removing service is in the load list, then, when you apply the next PUT tape, the correct text deck, including only the updates that you didn't comment out in the HCPABC AUXCOR file, is automatically used. Continue with Step 7 on page 584.
   - If the text deck for which you are removing service is not in the load list, you must determine which text deck (TXT*nnnnn*, where *nnnnn* is the PTF number) replaces the current HCPABC TXT*nnnnn* (which contains all four fixes, three of which you want to remove).

     a. First list all text decks for the module hit by the updates you want to remove:

```
filelist hcpabc txt* * ■


HCPABC   TXT00000 fm
HCPABC   TXT00001 fm
HCPABC   TXT00002 fm
HCPABC   TXT00003 fm
```

     b. Next, find the text deck which contains only the update(s) you want to keep, in this case HCPABC H12345HP. It should be the text deck corresponding to the first update you did not comment out of the auxiliary control file in Step 4 on page 582 (in this case, HCPABC TXT00000); but, to make sure, you can examine the text deck corresponding to the update at the top of the auxiliary control file (in this case, HCPABC TXT00003).

```
xedit hcpabc txt00003 ■
```

```
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
H12345HP  COR  UM00000  * Comments
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
H34567HP  COR  UM00001  * Comments
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
H56789HP  COR  UM00002  * Comments
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
* DEPEND: H01010HP
H88888HP  COR  UM00003  * Comments
:
(Executable text is here.)
```

HCPABC TXT00003 contains the three updates you want to delete and the one update you want to keep.

Find the text deck line which contains the latest update you want to keep. This is the update immediately above the first one you want to remove. The third word of that line is the PTF number for this update: UM00000. Prefix TXT to the numeric part of this PTF number for the file type of the new text deck: TXT00000.

c. Since the current text deck for HCPABC contains the fixes you want to delete, you must replace it with the text deck that contains only update H12345HP. Enter:

```
copyfile hcpabc txt00000 fm hcpabc text = (replace ■
Ready; T=n.nn/n.nn hh:mm:ss
```

fm is the file mode of the alternate LOCAL1 disk.

7. Next, check each update file that you deleted or commented out of the auxiliary control file for DEPEND entries. The DEPEND entries identify previous updates that are dependent on the update you are removing. Such updates must also be removed.

   In this example, suppose that only HCPABC H88888HP contains a DEPEND entry.

```
xedit HCPABC H88888HP ■
./* PREREQ: NONE
./* CO-REQ: NONE
./* IF-REQ: NONE
./* DEPEND: H01010HP
qquit ■
```

HCPABC H88888HP contains one DEPEND entry, for HCPABC H01010HP. (The file name of the dependent update file is the same as the file name of the update file you are looking at. The file type is listed in the DEPEND entry.)

8. Add any updates you find in DEPEND entries to your list of updates that you must remove. You know the APAR number for these updates (in this example, 01010) from the five numeric characters in the file type; but you do not yet know the corresponding PTF number. (Appendix I, "How To Find the PTF Number From the APAR Number" on page 865 will show you how to find it.) For now, put question marks in your delete list beside the DEPEND updates and come back to this point later.

The list of updates to be deleted now looks like this:

```
HCPABC H34567HP   PTF UM00001
HCPXYZ H34567HP   PTF UM00001
HCPABC H88888HP   PTF UM00003
HCPABC H56789HP   PTF UM00002
HCPXYZ H99999HP   PTF UM00004
HCPABC H01010HP   PTF ???????
```

9. Rename the update(s) you want to remove with a file mode of 1.

```
rename HCPABC H34567HP fm5 = = fm1 ■
Ready; T=n.nn/n.nn hh:mm:ss
rename HCPXYZ H34567HP fm5 = = fm1 ■
Ready; T=n.nn/n.nn hh:mm:ss
rename HCPABC H88888HP fm5 = = fm1 ■
Ready; T=n.nn/n.nn hh:mm:ss
rename HCPABC H56789HP fm5 = = fm1 ■
Ready; T=n.nn/n.nn hh:mm:ss
rename HCPXYZ H99999HP fm5 = = fm1 ■
Ready; T=n.nn/n.nn hh:mm:ss
rename HCPXYZ H01010HP fm5 = = fm1 ■
Ready; T=n.nn/n.nn hh:mm:ss
```

10. Repeat Steps 2 on page 581 through 9 for each update in your delete list. When you have done this, check the parts lists for the PTFs you noted in Step 5 on page 583 and delete any updates included in those PTFs that you missed. Then determine the PTF numbers for the DEPEND entries you found in Step 7 on page 584 (see Appendix I, "How To Find the PTF Number From the APAR Number" on page 865) and delete those PTFs.

11. Finally, rework local service to modules for which you have deleted service. (Local service is service listed at any AUX levels above the AUXCOR level in the control file for this component. See Chapter 18, "Receiving and Applying Local Service" on page 565.)

MAP 0017: What to Do Next

001

Do you have a service tape?
Yes **No**

002

– Go to the instructions for rebuilding the component from which you removed service:

CMS Chapter 13, "Rebuilding CMS after Applying Service" on page 457
CP Chapter 14, "Rebuilding CP after Applying Service" on page 515
DV Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility," "Step 4. Build the Dump Viewing Facility" on page 542
GCS Chapter 16, "Program Update Service or Corrective Service to GCS," "Step 4. Build a New GCS Macro Library" on page 552.

003

– Go to the instructions for receiving service for the component from which you removed service:

CMS Chapter 9, "Receiving Program Update Service or Corrective Service for CMS" on page 435
CP Chapter 10, "Receiving Program Update Service or Corrective Service for CP" on page 443
DV Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility," "Step 2. Receive Service" on page 539
GCS Chapter 16, "Program Update Service or Corrective Service to GCS," "Step 2. Receive Service" on page 549.

Part 3. Appendices

This book contains the following appendices:

- Appendix A, "VM/XA System Product Regeneration Requirements" on page 589, which shows which modules you must regenerate when you apply a fix to CMS code
- Appendix B, "EXEC and Command Format Summaries" on page 599, which shows the format and use of EXECs and commands you may need for installing and servicing VM/XA System Product
- Appendix C, "VM/XA System Product Starter System Information" on page 747, which gives sample system definition files and other information about the VM/XA SP starter system
- Appendix D, "Listing CP Data Areas and Control Blocks" on page 849, which describes a procedure for listing VM/XA System Product data areas and control blocks
- Appendix E, "Example of Alternate GCS Nucleus Placement" on page 853, which shows how to save your GCS nucleus at a different location than the one in the sample files
- Appendix F, "Restricted Logon Passwords" on page 857, which contains the IBM-supplied list of restricted passwords
- Appendix G, "Controlling Disk String Merges" on page 859, which tells you how to prevent the automatic merging of a disk string and how to merge disk strings manually
- Appendix H, "Service Reference Tables" on page 861, which lists the standard 3-character abbreviations for the various file types used in VM/XA SP and the parts supplied for service
- Appendix I, "How To Find the PTF Number From the APAR Number" on page 865, which shows the procedure for determining the number of a program temporary fix from the associated APAR number
- Appendix J, "Messages" on page 867, which contains messages issued by the service EXECs.

Appendix A. VM/XA System Product Regeneration Requirements

CMS Regeneration Requirements

Whenever you apply a fix to CMS source code, you must regenerate the CMS nucleus or some CMS modules. The following table shows which you must regenerate in each case. (If a source name does not appear in the table, either the file is contained within the CMS nucleus, or it is loaded by another file, for example, DMSBTB loads DMSBTP.)

If you must regenerate the CMS nucleus, see Chapter 13, "Rebuilding CMS after Applying Service" on page 457.

Table 15 (Page 1 of 6). CMS Regeneration Requirements

Change in Source	Requires Regeneration of Module/Nucleus	EXEC Procedure to Use
DMSACC	Nucleus	VMFBLD
DMSACF	Nucleus	VMFBLD
DMSACM	Nucleus	VMFBLD
DMSAMS	AMSERV	CMSGEND
DMSASD	ASSEMBLE	CMSGEND
DMSASM	ASSEMBLE	CMSGEND
DMSASN	ASSGN	CMSGEND
DMSBAB	CMSDOS	DOSGEN
DMSBCT	DMSDFT	CMSGEND
DMSBLG	DMSDFT	CMSGEND
DMSBOF	DMSCUT	CMSGEND
DMSBOP	CMSDOS	DOSGEN
DMSBTB	CMSBATCH	CMSGEND
DMSBUS	DMSCUT	CMSGEND
DMSCCM	DMSCUT	CMSGEND
DMSCDI	DMSDFT	CMSGEND
DMSCCK	CATCHECK	CMSGEND
DMSCIA	DMSCUT	CMSGEND
DMSCLS	CMSDOS	DOSGEN
DMSCMP	COMPARE	CMSGEND
DMSCPY	Nucleus	VMFBLD
DMSCVH	CMSDOS	DOSGEN
DMSDAS	CMSDOS	DOSGEN

Table 15 (Page 2 of 6). CMS Regeneration Requirements

Change in Source	Requires Regeneration of Module/Nucleus	EXEC Procedure to Use
DMSDFT	DMSDFT	CMSGEND
DMSDLK	DOSLKED	CMSGEND
DMSDMP	CMSDOS	DOSGEN
DMSDOS	CMSDOS	DOSGEN
DMSDSK	DISK	CMSGEND
DMSDSL	DOSLIB	CMSGEND
DMSDSV	DSERV	CMSGEND
DMSEDC	EDIT (see Note 1 on page 594)	CMSGEND
DMSEDF	EDIT (see Note 1 on page 594)	CMSGEND
DMSEDI	EDIT (see Note 1 on page 594)	CMSGEND
DMSEDX	EDIT (see Note 1 on page 594)	CMSGEND
DMSEND	CMSAMS	VSAMGEN
DMSEXG	DCSSGEN	CMSGEND
DMSEXM	EXECMAP	CMSGEND
DMSFCH	CMSDOS	DOSGEN
DMSFOR	FORMAT	CMSGEND
DMSGIO	EDIT (see Note 1 on page 594)	CMSGEND
DMSGLB	GLOBAL	CMSGEND
DMSGLO	Nucleus	VMFBLD
DMSGMF	CMSDOS	DOSGEN
DMSGND	GENDIRT	CMSGEND
DMSGTM	CMSDOS	DOSGEN
DMSGVE	CMSDOS	DOSGEN
DMSHLB	HELPCONV	CMSGEND
DMSHLD	HELPCONV	CMSGEND
DMSHLI	HELPCONV	CMSGEND
DMSHLP	HELPCONV	CMSGEND
DMSHLS	HELPCONV	CMSGEND
DMSICP	IOCP	CMSGEND
DMSIDE	Nucleus	VMFBLD
DMSICT	DMSCUT	CMSGEND
DMSIMA	IMAGEMOD	CMSGEND
DMSLAB	CMSDOS	DOSGEN
DMSLBD	LABELDEF	CMSGEND

Table 15 (Page 3 of 6). CMS Regeneration Requirements

Change in Source	Requires Regeneration of Module/Nucleus	EXEC Procedure to Use
DMSLBM	MACLIB	CMSGEND
DMSLBT	TXTLIB	CMSGEND
DMSLCK	CMSDOS	DOSGEN
DMSLDF	CMSDOS	DOSGEN
DMSLDS	LISTDS	CMSGEND
DMSLIC	CMSDOS	DOSGEN
DMSLLU	LISTIO	CMSGEND
DMSLMX	TAPE (see Note 2 on page 594)	CMSGEND
DMSMCM	CMSDOS	DOSGEN
DMSMDP	MODMAP	CMSGEND
DMSMES[*y*]	DMSMES[*y*] (see Note 3 on page 594)	VMFNLS
DMSMGC	GENMSG	CMSGEND
DMSMGD	GENMSG	CMSGEND
DMSMGE	GENMSG	CMSGEND
DMSMVE	MOVEFILE	CMSGEND
DMSOPL	CMSDOS	DOSGEN
DMSOR1	CMSDOS	DOSGEN
DMSOR2	CMSDOS	DOSGEN
DMSOR3	CMSDOS	DOSGEN
DMSNXD	NUCXDROP	CMSGEND
DMSOPT	OPTION	CMSGEND
DMSOSR	OSRUN	CMSGEND
DMSOVR	SVCTRACE	CMSGEND
DMSOVS	DMSOVS	CMSGEND
DMSPIO	Nucleus	VMFBLD
DMSPBS	DMSCUT	CMSGEND
DMSPCA	DMSPCC	CMSGEND
DMSPCB	DMSPCC	CMSGEND
DMSPCC	DMSPCC	CMSGEND
DMSPCR	DMSPCC	CMSGEND
DMSPCT	DMSPCC	CMSGEND
DMSPCW	DMSPCC	CMSGEND
DMSPOA	PROPLIB (see Note 4 on page 594)	CMSGEND

Table 15 (Page 4 of 6). CMS Regeneration Requirements		
Change in Source	**Requires Regeneration of Module/Nucleus**	**EXEC Procedure to Use**
DMSPOC	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPOD	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPOE	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPOL	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPON	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPOP	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPOQ	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPOR	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPOS	PROPLIB (see Note 4 on page 594)	CMSGEND
DMSPRE	PRELOAD	CMSGEND
DMSPRT	Nucleus	VMFBLD
DMSPRV	PSERV	CMSGEND
DMSPUN	PUNCH	CMSGEND
DMSRDC	READCARD	CMSGEND
DMSRDR	RDR	CMSGEND
DMSRNE	RENUM	CMSGEND
DMSRPG	CMSDOS	DOSGEN
DMSRRV	RSERV	CMSGEND
DMSRSV	RESERVE	CMSGEND
DMSSAP	DMSCUT	CMSGEND
DMSSCR	EDIT (see Note 1 on page 594)	CMSGEND
DMSSFD	SAVEFD	CMSGEND
DMSSMG	DMSDFT	CMSGEND
DMSSNC	DMSSNC	CMSGEND
DMSSPA[*y*]	DMSSPA[*y*] (see Note 3 on page 594)	VMFNLS
DMSSPR	SETPRT	CMSGEND
DMSSRT	SORT	CMSGEND
DMSSSK	SETKEY	CMSGEND

Table 15 (Page 5 of 6). CMS Regeneration Requirements		
Change in Source	**Requires Regeneration of Module/Nucleus**	**EXEC Procedure to Use**
DMSSTX	CMSDOS	DOSGEN
DMSSUB	CMSDOS	DOSGEN
DMSSUP	DMSCUT	CMSGEND
DMSSYN	SYNONYM	CMSGEND
DMSSVL	CMSDOS	DOSGEN
DMSTMA	TAPEMAC	CMSGEND
DMSTPD	TAPPDS	CMSGEND
DMSTPE	TAPE (see Note 2 on page 594)	CMSGEND
DMSTPF	TAPE (see Note 2 on page 594)	CMSGEND
DMSTPG	TAPE (see Note 2 on page 594)	CMSGEND
DMSTPH	TAPE (see Note 2 on page 594)	CMSGEND
DMSTPI	TAPE (see Note 2 on page 594)	CMSGEND
DMSTPJ	TAPE (see Note 2 on page 594)	CMSGEND
DMSTRC	DMSCUT	CMSGEND
DMSTRT[*y*]	DMSTRT[*y*] (see Note 3 on page 594)	VMFNLS
DMSTYP	TYPE	CMSGEND
DMSUPD	UPDATE	CMSGEND
DMSUSR	DMSDFT	CMSGEND
DMSUTL	LOADLIB	CMSGEND
DMSVAN	CMSAMS	VSAMGEN
DMSVAS	CMSAMS	VSAMGEN
DMSVAX	CMSAMS	VSAMGEN
DMSVIP	CMSVSAM	VSAMGEN
DMSVIS	CMSDOS	DOSGEN
DMSVLT	CMSDOS	DOSGEN
DMSVVN	CMSAMS	VSAMGEN
DMSVVS	CMSAMS	VSAMGEN
DMSXCP	CMSDOS	DOSGEN
DMSXMS	DMSXMS (see Note 5 on page 594)	CMSGEND
DMSXRE	DMSXRE (see Note 5 on page 594)	CMSGEND
DMSZAP	ZAP	CMSGEND
DMSZIT	EDIT (see Note 1 on page 594)	CMSGEND

Table 15 (Page 6 of 6). CMS Regeneration Requirements

Change in Source	Requires Regeneration of Module/Nucleus	EXEC Procedure to Use
IOPC*xxxx*	IOCP	CMSGEND
IOPP*xxxx*	IOCP	CMSGEND
VMFCLEAR	VMFCLEAR	GENMOD
VMFDATE	VMFDATE	CMSGEND
VMFDOS	VMFDOS	CMSGEND
VMFLOAD	VMFLOAD	CMSGEND

Notes:

1. When invoked for EDIT, the CMSGEND EXEC procedure creates the EDIT module and then reinvokes itself to create the EDMAIN module.
2. When the CMSGEND EXEC is invoked for TAPE, it creates the TAPE module and then reinvokes itself to create the DMSLMX and DMSTP*x* modules.
3. *y* is the country code for your system national language. See Table 5 on page 317.
4. You get messages DMSSLK0008W and DMSSOP036E when you regenerate PROPLIB. You can ignore them.
5. All EDIT source files, except DMSXMS and DMSXRE, are contained within the CMS nucleus.

CP Regeneration Requirements

If you apply corrective service to certain CP source files, you must also regenerate the corresponding CMS modules and stand-alone utilities that can be IPLed. The following table shows the source name, module name, and procedures used for regenerating the CMS module.

The UTILITY EXEC is described in Appendix B, "EXEC and Command Format Summaries" on page 599.

Table 16. CP Regeneration Requirements

Change in Source	Requires Regeneration of Module/Nucleus	EXEC Procedure(s) to Use
HCPBSL	3CARD LOADER	UTILITY
HCPCCF HCPFAA-H, HCPFAL-M, HCPFAR, HCPFON	CPFMTXA	UTILITY
HCPCCU	HCPCCU	UTILITY
HCPDDC, HCPDDR, HCPDDT, HCPDNC, HCPDNT	IPL DDRXA	UTILITY
HCPDIR	DIRECTXA	UTILITY
HCPED*x*	DUMPLOAD	UTILITY
HCPIFC, HCPREA	CPEREPXA	UTILITY
HCPIMG	GENIMAGE	UTILITY
HCPLDR	HCPLDR	UTILITY
HCPLDL, HCPMDLAT	CPLOAD EXEC	UTILITY
HCPMES[*y*]	HCPMES[*y*]	VMFNLS
Note: *y* is the country code for your system national language. See Table 5 on page 317.		
HCPNMT	IMAGELIB	UTILITY
HCPOPTNS	HCPXA, HCPLDRCM	VMFMAC
HCPOVE	OVERRIDE	UTILITY
HCPRET	RETRIEVE	UTILITY
HCPSADMP	HCPSADMP	UTILITY

Dump Viewing Facility Regeneration Requirements

If you apply corrective service to the dump viewing facility TEXT files, you must also regenerate the dump viewing facility. The following table shows the TEXT file name, and the procedure for regenerating the dump viewing facility.

Table 17. Dump Viewing Facility Regeneration Requirements

Change in TEXT file	Requires Regeneration of Module	EXEC Procedure(s) to Use
HCS*xxx*	ADDMAP, DUMPSCAN, MAP, PRTDUMP, SCAN, TRACERED	UTILITY

GCS Regeneration Requirements

If you have local modifications to the GCS message repository, you must regenerate it. The following table shows the file name and the procedure to be used for regenerating the dump viewing facility.

Table 18. GCS Regeneration Requirements

Change in Source	Requires Regeneration of Module	EXEC Procedure(s) to Use
CSIMES[*y*]	CSIMES[*y*]	VMFNLS

Note: *y* is the country code for your system national language. See Table 5 on page 317.

Appendix B. EXEC and Command Format Summaries

This section is a general reference for commands and EXECs you may use during system generation or service application. The commands and EXECs that appear here are:

- ASMGEND EXEC
- CMSGEND EXEC
- DCSSGEN command
- DIRECTXA command
- DISKMAP EXEC
- DOSGEN EXEC
- GROUP EXEC
- HCPLDR command
- INSTFPP EXEC
- ITASK EXEC
- The Patch Facility
- PRELOAD MODULE
- SAMGEN EXEC
- SAMPNSS EXEC
- SETUP EXEC
- SPLOAD EXEC
- UPDATE command
- UTILITY EXEC
- VMFAPPLY EXEC
- VMFBLD EXEC
- VMFHASM EXEC
- VMFLKED EXEC
- VMFMAC EXEC
- VMFMERGE EXEC
- VMFNLS EXEC
- VMFOVER EXEC
- VMFPLC2 command
- VMFREC EXEC
- VMFREMOV EXEC
- VMFSETUP EXEC
- VMFVIEW EXEC
- VMFZAP EXEC
- VSEVSAM EXEC
- ZAP MODULE
- ZAPTEXT EXEC.

ASMGEND EXEC

The ASMGEND EXEC procedure builds the system assembler and creates the associated auxiliary directory. ASMGEND loads the text decks for the assembler in the correct overlay structure and produces a load map.

Format

The format of the ASMGEND EXEC is:

ASMGEND	

Usage Notes

1. The assembler text decks normally reside on the system S-disk in file mode S1. This disk must be accessed in some additional file mode before issuing this command in order to locate these files. For example:

   ```
   access 190 a ■
   access 193 b ■
   ```

2. Use the ASMGEND EXEC if you have modified the assembler (IFOX*nn* MODULE) source. If you have not modified this source, and wish to create the assemble module, possibly after modifying DMSNGP or after creating a new CMS system disk, use the CMSGEND EXEC.

Messages

ENTER TARGET DISK MODE FOR ASSEMBLE MODULES
DEFAULTS TO S-DISK IF NONE ENTERED.

Explanation: Enter the mode letter of the disk containing the assembler modules. The ASSEMBLE command accesses this disk during processing. If you enter a mode letter, ASMGEND uses that mode letter as the "targetmode" operand of the GENDIRT command when it creates the auxiliary directory. If you do not specify a mode letter, S is used.

ASSEMBLE XF GEND COMPLETE.

Explanation: The system assembler and its associated auxiliary directory have been generated successfully.

ASSEMBLE XF GEND FAILED.

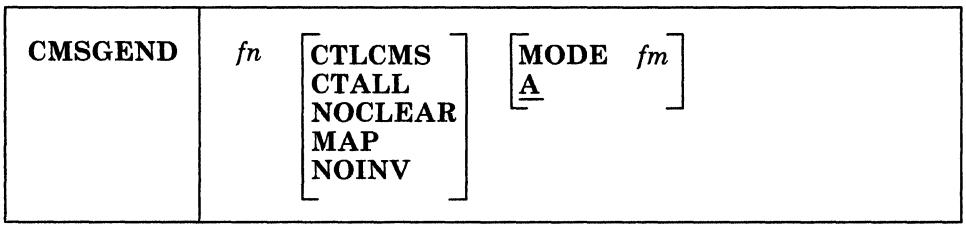
Explanation: The system assembler text files were not loaded successfully.

CMSGEND EXEC

Use the CMSGEND EXEC procedure to generate a new CMS module or LOADLIB from a text file and place the new CMS module or LOADLIB on the specified disk.

Format

The format of the CMSGEND command is:



where:

fn
is the file name of the CMS module or LOADLIB that is to be generated by the CMSGEND EXEC. Only one file name may be specified in the CMSGEND command line.

The file names that may be specified in the CMSGEND command are any disk-resident CMS commands and service programs.

CTLCMS
displays each CMS command as it is executed in the CMSGEND EXEC procedure. This is equivalent to the EXEC statement &CONTROL CMS.

CTALL
displays every executable statement as it is executed in the CMSGEND EXEC procedure. This is equivalent to the EXEC statement &CONTROL ALL.

NOCLEAR
specifies that the CLEAR option is not to be issued when CMSGEND invokes the LOAD command.

MAP
specifies that the NOMAP option is not to be issued when CMSGEND invokes the GENMOD command.

NOINV
issues the NOINV option when CMSGEND invokes the LOAD command; this suppresses the displaying of invalid cards at the terminal. If the text deck was created with the VMFASM EXEC, it may contain update listing information; these records are displayed during the loading process unless you specify NOINV.

MODE *fm*
indicates the access mode of the disk to receive the new module. File mode A is the default.

Usage Notes

1. The assembler text decks normally reside on the system S-disk in file mode S1. This disk must be accessed in some additional file mode before you issue this command in order to locate these files. For example:

```
access 190 a ■ (CMS system disk)
access 293 b ■ (CMS DELTA disk)
access 193 c ■ (CMS BASE disk)
```

2. Use the CMSGEND EXEC if you have not modified the assembler (IFOX*nn*) source, and wish to create the assemble module, possibly after modifying DMSASM, DMSASN, DMSASD, or creating a new CMS system disk.
3. You can also use the CMSGEND EXEC to regenerate the ASSEMBLE command when you move the CMS system disk. When you specify ASSEMBLE, CMSGEND prompts you to enter a disk mode letter so it can refresh the assembler's auxiliary directory. (Use the ASMGEND EXEC procedure if you are updating the assembler.)
4. When using CMSGEND EXEC to regenerate the PROP command, you are only generating the PROPLIB LOADLIB. (The CMSGEND options NOCLEAR, MAP, and NOINV have no effect when generating the PROP command.)
5. The file type of the text deck must be TEXT.

How CMSGEND Works

CMSGEND keeps a list of the CMS disk-resident modules and LOADLIBs, the file names of the text files used to create them, and any special attributes required to generate them. For example, the PRINT command must be generated with the ORIGIN TRANS and the SYSTEM options. It is composed of the DMSPRT text file. To generate a new PRINT module, you issue:

```
cmsgend print ■

*** CURRENT STATUS:
FILE ' PRINT MODULE A2 ' DOES NOT EXIST
FILE ' PRINT MODULE A1 ' DOES NOT EXIST

*** LOADING:
 INVALID CARD - *  DMSSP  MACLIB  A1 5.5CMS mm/dd/yy hh:mm

:
 DMSPRT     SO 00E000
 PRINT         00E000

 ***RESULTS:
 ' PRINT MODULE A2 ' CREATED FROM TEXT DECK ( S ) DMSPRT

 WITH OPTIONS TRAMX SYSTEM NOMAP

 R;
```

Messages

The CMSGEND EXEC procedure displays the following status and error messages:

***** CURRENT STATUS: "*fn* MODULE A*n*" [EXISTS|DOES NOT EXIST].**

Explanation: This message indicates whether a generated module already exists.

***** LINK EDITING: *fn* TEXT.**

Explanation: This message indicates that CMSGEND is link editing *fn* TEXT to create a LOADLIB. The existing LOADLIB is erased and not renamed when generating a new one.

***** LOADING:**

Explanation: This message indicates that CMSGEND is loading the text decks.

***** (UNDEF. NAMES NORMAL FOR EDMAIN)**
***** NOW WE HAVE A SECOND PASS FOR EDMAIN MODULE.**

Explanation: These messages indicate that the EDIT command requires two passes to resolve undefined names.

***** NOW WE HAVE A SECOND PASS FOR DMSTPI MODULE.**

Explanation: This messages indicates that the TAPE command requires two passes to generate the necessary modules. The first pass generates the DMSLMX module as the TAPE module; the second pass generates the DMSTPE, DMSTPG, DMSTPH, DMSTPI, and DMSTPJ text files as the DMSTPI module.

***** RESULTS:**
['*fn*' MODOLD A1' WAS ERASED]
['*fn* MODULE A2' RENAMED TO '*fn* MODOLD A1']

'*fn* MODULE A2' CREATED FROM TEXT DECK(S) ...
WITH OPTIONS ...

Explanation: These messages indicate which existing modules were erased and renamed, which text files were used to create the new module, and the attributes used to create the module.

ENTER GENDIRT TARGET DISK MODE LETTER
(NULL LINE DEFAULTS TO "S" DISK).

Explanation: This message is issued when you specify ASSEMBLE. You should enter the mode letter of the disk that contains the system assembler. This letter is used as the target disk mode address for the GENDIRT command.

***** ERROR MESSAGE ISSUED IS NORMAL FOR LINK EDITING.**

Explanation: If the TEXT deck was created with VMFASM EXEC, it may contain update listing information; these records will cause the linkage editor to generate an error message. The error is normal.

ERROR OCCURRED. CMSGEND STOPS.

Explanation: This message indicates that an error occurred and that CMSGEND processing ends.

INVALID ARGUMENT *fn*.

Explanation: This message indicates an invalid file name was specified on the command line.

TYPE 'CMSGEND *fn* < options > '.

Explanation: This message indicates that no file name was specified on the command line.

DCSSGEN Command

To install CMSINST, use the DCSSGEN command procedure. This command lets you build, load, and save a CMSINST segment that contains the EXECs and Editor macros that you select for your installation.

The installation segment (CMSINST is the default name) is designed to contain EXECs and System Product Editor (XEDIT) macros. When a frequently used EXEC or editor macro resides in a saved segment, multiple users can share the same executing copy.

Before you issue the DCSSGEN command you must create a file that contains a load list of the EXECs and System Product Editor (XEDIT) macro instructions to be loaded into the segment. Information about the load list follows this description of the DCSSGEN command.

Format

The format of the DCSSGEN command is:

DCSSGEN	*filename filetype filemode* [*segname* / CMSINST]

where:

filename filetype filemode
is the file ID of your load list.

[*segname*]
[**CMSINST**]

[***segname***
CMSINST]

is the name that you want to assign to the saved segment (*segname*). The default segment name is CMSINST.

Note: The values for this command are positional.

DCSSGEN performs the following operations:

- Processes the loadlist file, sequentially loading each EXEC and Editor macro into storage
- Saves the segment
- Writes a load map to the A-disk as *segname* DCSSMAP A.

The Load List for the DCSSGEN Command

The loadlist must be a fixed-format file with a logical record length of 80. Each record in the file must contain the fileid of one EXEC or System Product Editor macro or a comment. DCSSGEN processes the records sequentially.

The format of a DCSSGEN loadlist entry is:

```
fn  ft  [fm [execname  [exectype] ] ]
```

where:

fn
is the file name of the EXEC or editor macro to be loaded.

ft
is the file type of the EXEC or editor macro to be loaded.

fm
is the file mode of the EXEC or editor macro to be loaded. If the file mode is specified as *, DCSSGEN loads the first file in the disk search order that satisfies the file name and file type qualifications.

execname
is the file name to be assigned to the loaded EXEC or editor macro. The default is "=", which means that the present file name is to be used.

exectype
is the file type to be assigned to the loaded EXEC or editor macro. The default is "=", which means that the present file type is to be used.

The file name and file type of the EXEC or editor macro can each be from one to eight characters. The valid characters are A-Z, a-z, 0-9, $, #, @, +, - (hyphen), : (colon), and _ (underscore). The execname and exectype may also be from one to eight characters. However, they are not limited to the file name and file type character set. The only characters NOT valid within an execname and exectype are =, *,) (right parenthesis), ((left parenthesis), and X'FF'.

To enter a comment in the loadlist, type an asterisk (*) in column one followed by the text of the comment.

For example, your loadlist entries may look like this sample:

```
* Rename RDRLIST EXEC to MAIL EXEC
RDRLIST EXEC * MAIL =
FILELIST EXEC S
SYSPROF EXEC S
PARSE XEDIT S
DISCARD EXEC S
NOTE EXEC S
PROFNOTE XEDIT S
ALL XEDIT S
```

Before you process your loadlist, remove the comments and unnecessary blanks from the source program to conserve storage space. The EXECUPDT command with the NOCOMMENTS option removes all comments and leading blanks. One comment line containing the exec name and exec type is inserted at the beginning of the file. If the source file contains Double-Byte Character Set (DBCS) characters, also specify the ETMODE option. For more information about the EXECUPDT command, refer to *VM/XA SP CMS Command Reference*.

In the load map file, the records copied from your loadlist file are left-justified. The records created during the build process are indented five spaces. Comments are also copied from your loadlist file, with an asterisk (*) in column one followed by the text.

The load map file (CMSINST DCSSMAP A) for the sample loadlist would look like this:

```
* RENAME RDRLIST EXEC TO MAIL EXEC
RDRLIST EXEC * MAIL =
    15:41:59 10/22/85 copy of RDRLIST  EXEC     S loaded as MAIL     EXEC
    EXISBLK - 280000   FBLOCK - 280100   LENGTH - 001C40
FILELIST EXEC S
    15:41:58 10/22/85 copy of FILELIST EXEC     S loaded as FILELIST EXEC
    EXISBLK - 280020   FBLOCK - 281D40   LENGTH - 0018C8
SYSPROF EXEC S
     7:30:18 11/26/85 copy of SYSPROF  EXEC     S loaded as SYSPROF  EXEC
    EXISBLK - 280040   FBLOCK - 283608   LENGTH - 002178
PARSE XEDIT S
     8:47:55 12/18/84 copy of PARSE    XEDIT    S loaded as PARSE    XEDIT
    EXISBLK - 280060   FBLOCK - 285780   LENGTH - 0024A0
DISCARD EXEC S
    15:40:32 10/22/85 copy of DISCARD  EXEC     S loaded as DISCARD  EXEC
    EXISBLK - 280080   FBLOCK - 287C20   LENGTH - 0012B0
NOTE EXEC S
    11:03:53 10/24/85 copy of NOTE     EXEC     S loaded as NOTE     EXEC
    EXISBLK - 2800A0   FBLOCK - 288ED0   LENGTH - 005310
PROFNOTE XEDIT S
    15:41:55 10/22/85 copy of PROFNOTE XEDIT    S loaded as PROFNOTE XEDIT
    EXISBLK - 2800C0   FBLOCK - 28E1E0   LENGTH - 000980
ALL XEDIT S
    15:41:05 10/22/85 copy of ALL      XEDIT    S loaded as ALL      XEDIT
    EXISBLK - 2800E0   FBLOCK - 28EB60   LENGTH - 001298
    *** End of Source List ***
    CMSINST built at 15:56:34 on 12/02/86
```

DIRECTXA Command

DIRECTXA is a CMS command used to create a CP user directory.

Note: This command works in a System/370 mode virtual machine only.

Format

DIRECTXA	[*fn* [USER] [*ft* [DIRECT] [*fm* [*] [–]]]] [(*options* ... [)]] *options*: [EDIT] [MIXED]

where:

[*fn* **USER**]

is the file name of the directory. If not specified, the file name is USER.

[*ft* **DIRECT**]

is the file type of the directory. If not specified, the file type is DIRECT.

[*fm* * –]

is the file mode of the directory. If not specified, the file type is * (the command searches all accessed disks for the file name and file type).

EDIT
allows you to issue the DIRECTXA command without updating the directory on disk. With this option, you can check the syntax of directory control statements.

MIXED
specifies that the DIRECTXA command is to ignore VM/SP statements and options encountered in the directory. This allows you to use a directory from a VM/SP system. DIRECTXA prints informational messages for each statement or option it ignores.

How the Directory Program Works

The DIRECTXA command performs the following steps during execution:

1. The program looks for the file you specified on the DIRECTXA command line. If you did not specify a file name or a file type, the program looks for a file that has the file name of USER and a file type of DIRECT.
2. If the program does not find the directory or an error occurs during processing, the program does not create a directory and the old directory remains on line. The program will continue to check the syntax of all control statements before ending.

3. DIRECTXA looks for the RPWLIST DATA file, which contains a list of restricted passwords. (The RPWLIST DATA file supplied by IBM is printed in Appendix F, "Restricted Logon Passwords" on page 857. You can edit it to add your own restricted passwords.) If the RPWLIST DATA file is not found, DIRECTXA issues a warning message but continues processing. If the RPWLIST DATA file is found, DIRECTXA checks all the passwords in the directory against it and changes any restricted passwords (except the password of the user issuing the DIRECTXA command) to NOLOG. Any user whose password is changed to NOLOG will not be able to log on. If the password of the virtual machine issuing the DIRECTXA command is restricted, it is not changed to NOLOG. DIRECTXA issues an error message and does not update the directory.
4. If DIRECTXA runs out of DASD space, the program prints an error message and ends immediately. To continue, you must allocate more cylinders for the user directory.

   If DIRECTXA runs out of virtual storage, the command issues an error message and ends immediately. To continue, you must define more virtual storage (DEFINE STORAGE command). Using the following formula, calculate the size of virtual storage you need.

   vstor = 50 Kb + (16 bytes × *userids*)

   where:

   vstor
   is the amount of virtual storage in the program area that DIRECTXA requires.

   userids
   is the total number of users defined in the directory.
5. If no errors occur, the program writes the directory on DASD and writes the DASD address of the new directory on the volume label for that device. If the program is updating an active system directory, the program makes the new directory immediately available for the system.

Usage Notes

1. A complete description of the directory program, including a description of directory control statements, is contained in *VM/XA SP Planning and Administration.*
2. When writing a user directory, DIRECTXA does not overwrite the current directory, but does write an alternate user directory. The directory pointer in cylinder 0 is then changed to reference this new directory. It is advisable, therefore, to issue the DIRECTXA command twice. This assures that the directory you wish to use is available; otherwise, you may lose your directory again.

Restrictions

1. The DIRECTXA command executes only under CMS.
2. You must have CP privilege class A, B, or C in order to update the system user directory.
3. You cannot use the DIRECTXA command to create a directory in VM/XA SP Release 2 format on the VM/XA SP Release 1 system residence volume. You must create the Release 2 directory on a different physical pack.

Examples

To compile the USER DIRECT source directory and apply it to the system if successful, enter:

```
directxa user direct * ■
```

To do a test compile of a USER DIRECT source directory from a VM/SP HPO 5 system, ignoring control statements that are valid only for VM/SP or that contain options valid only for VM/SP, enter:

```
directxa user direct * (edit mixed ■
```

Messages

VM/XA SYSTEM PRODUCT USER DIRECTORY CREATION PROGRAM - RELEASE 2.0.

Explanation: You have invoked the directory program.

EOJ DIRECTORY UPDATED AND ON LINE.

Explanation: The directory has been updated. The new directory has been placed in use by CP.

EOJ DIRECTORY UPDATED.

Explanation: The directory has been updated, but has not been placed in use by CP. You receive this response if you write the directory to a CP-formatted DASD that is not a CP-owned volume, or if you do not have the necessary privilege class to change the CP directory that is in use.

EOJ DIRECTORY NOT UPDATED.

Explanation: The directory has not been updated. You receive this response if you issue the DIRECTXA command with the EDIT option. You also receive this response if errors prevent the directory from being updated.

Return Codes

The DIRECTXA command issues the following return codes:

Return Code	Possible Causes
0	DIRECTXA executed successfully. The real CP directory has been updated (unless the EDIT option was specified).
1	The directory source file was not found on an accessed disk.
2	An error was encountered while processing the directory source file.
3	An invalid option was specified on the DIRECTXA command line.
4	No errors were encountered, but you do not have the proper privilege class to update the real CP directory.
5	Condition code 1 was received from the DIAGNOSE X'3C'. A class A, B, or C user updated a virtual directory.
6	Condition code 2 was received from the DIAGNOSE X'3C'. An invalid directory pointer was found on cylinder 0 record 3.
7	Condition code 3 was received from the DIAGNOSE X'3C'. A fatal I/O error occurred.
9	The directory has been rewritten, but warning messages have been issued.
> 100	Return codes greater that 100 may be returned and will be accompanied by message 764. See the explanation for message 764 for details on these return codes.
333	DIRECTXA was run in EDIT mode, and at least one invalid password was changed to NOLOG.

DISKMAP EXEC

The DISKMAP EXEC summarizes the MDISK statements in the user directory. The output produced by this EXEC shows gaps and overlaps between minidisk assignments.

Format

The format of the DISKMAP EXEC is:

DISKMAP	*filename*	[*filetype* / **DIRECT**]

where:

filename
is the file name of the directory to be mapped.

filetype
is the file type of the directory to be mapped. The default is DIRECT.

The output from the DISKMAP EXEC is a file sent to your A-disk. The file name of the output is the same as that of the target directory. The file type of the output is DISKMAP. The file contains information on MDISK statements found in the directory. The files are in order by volume in the output file. Gaps between minidisks and overlapping minidisks are flagged.

DISKMAP does NOT replace the EDIT function of the DIRECTXA command. You should use both to check your directory after changes. (For a description of the DIRECTXA command, see "DIRECTXA Command" on page 607.)

Usage Notes

1. Because some DASD types come in several sizes, DISKMAP does not list gaps found after all minidisks. You need to know the maximum cylinder/block value for your DASD type.
2. DISKMAP creates both the map and a workfile on your A-disk. If your directory is very large and your A-disk is almost filled, you may need to find some extra disk space in order to run DISKMAP.
3. You may choose to include some overlaps in the directory. DISKMAP flags *all* overlaps; you must understand your layout to determine if a particular overlap is expected or in error.

Example

To see how DISKMAP works, enter:

```
diskmap user ■
```

This produces a map of the sample directory that you loaded during installation of VM/XA SP.

DOSGEN EXEC

Use the DOSGEN EXEC to create the segment called CMSDOS, which contains text files needed to create a CMS/DOS environment that simulates DOS/VSE (Disk Operating System/Virtual Storage Extended) under CMS.

To install CMSDOS, use the DOSGEN EXEC procedure in Step 27 of Chapter 2, "Installing VM/XA System Product Release 2 with the Starter System (First Level)," Step 27 of Chapter 3, "Installing VM/XA System Product Release 2 with the Starter System (Second Level)," or Step 29 of Chapter 4, "Installing VM/XA System Product Release 2 Using an Existing VM/SP or VM/SP HPO System." You invoke DOSGEN with the hex load address and the name that you want to assign to this segment. This load address and name must match the address and name that you defined for this segment.

Notes:

1. The CMSDOS segment must be defined at a higher address than the CMSBAM segment.
2. Before you load and save CMSDOS, you must define your virtual machine with at least 512KB free storage above the end of the CMSDOS segment. This provides room for the loader tables, which occupy the top pages of virtual storage. After you load and save the segment, the loader tables and the 512KB free storage are no longer required.

DOSGEN performs the following operations:

- Checks that the specified virtual address contains valid characters and that it is greater than X'20000' and less than 16MB
- Looks for a read/write accessed A-disk on which to write the CMS loader work file
- Loads all the text files needed for VSE simulation, starting at the address specified
- Assigns a storage protection key of **X'D'**
- Saves the segment
- Writes the load map to the A-disk as LOAD MAP A5.

Messages

DMSGEN095E INVALID ADDRESS.

Explanation: DOSGEN detected an error in the address that you specified.

DMSGEN006E NO READ/WRITE A-DISK ACCESSED.

Explanation: DOSGEN could not find a read/write accessed A-disk.

DMSGEN111E DOSGEN FAILED DUE TO LOAD ERRORS.

Explanation: DOSGEN found unresolved external references while loading the text files.

DMSGEN412S DOSGEN FAILED DUE TO SETKEY ERRORS.

Explanation: DOSGEN detected an error while assigning the storage key.

DMSGEN141S DOSGEN FAILED DUE TO SAVESYS ERRORS.

Explanation: DOSGEN detected an error while saving the segment.

GROUP EXEC

Use the GROUP EXEC to set up a group control system (GCS) configuration file. You can invoke the GROUP EXEC from the CMS environment. When you do, you see a series of panels on which you enter information about your virtual machine group.

Format

The format of the GROUP EXEC is:

GROUP	[*systemname*]

where:

systemname
is an optional parameter that specifies the file name that you want to assign to the configuration file. If you enter this parameter with the GROUP command, then the Primary Option Menu panel appears with the SYSTEM NAME field filled in. If you enter the GROUP command without this parameter, then you must complete the SYSTEM NAME field on the Primary Option Menu panel. In either case, the system name that you specify must match the SYSNAME entry for this GCS system in the DMKSNT file. The sample SYSNAME entry is GCS.

For an example of using the GROUP EXEC panels, see "Step 25. Load, Build, and Save GCS" on page 75.

HCPLDR Command

The HCPLDR command invokes and controls the system loader to perform one of two possible functions:

- If you issue the HCPLDR command without the PUNCH option or the TAPE option, the loader generates a system (directly under CMS); this means the system loader link-edits all object files and passes control to the program (HCPGEN) that writes the system on the system residence device.
- If you issue the HCPLDR command with either the TAPE option or the PUNCH option, the loader creates a continuous card deck beginning with a three-card loader, continuing with a stand-alone version of the system loader, and ending with the object files belonging to the system to be generated. When you issue HCPLDR with the TAPE or PUNCH option to generate CP, the loader will not link-edit object modules and will not pass control to HCPGEN to write the system on the system residence device.

Issue the HCPLDR command under CMS.

Note: This command works in a System/370 mode virtual machine only.

Format

HCPLDR	loadlist [ctlfile / *ft*1[/][... *ft*256]] [(*options* ... [])]] *options:* [MAP] [PAGEB] [TAPE / PUNCH] [NOCTL]

where:

loadlist
is the file name of the load list EXEC file; it contains the names of the object files in the order in which they are to be loaded. The load list may look like:

```
&CONTROL OFF
&1 &2 &3 HCPLDR LOADER
&1 &2 &3 fn [ft]
&1 &2 &3 fn [ft]
⋮
```

fn and *ft* are the file name and file type of an object file. The *ft* is optional. If *ft* is omitted, HCPLDR searches the control file or an explicit list of file types to determine the file type.

HCPLDR LOADER must always be the first file. HCPLDR loads each object module in order from the top of the load list to the bottom.

You can place most loader control statements in the load list EXEC file. You cannot place VERIFY, REPLACE, or INCLUDE CONTROL SECTION statements in the EXEC file. A loader control statement begins with a 12-2-9 punch (X'02'). Whenever HCPLDR encounters X'02' in the first column of a record, it interprets that record as a control statement. For more information see "Loader Control Statements" on page 615.

ctlfile
is the file name of the control file. For more information about control files, see "UPDATE Command" on page 657.

HCPLDR uses the update level identifiers in the control file to determine the file type for those object modules listed in the load list without file types. The command skips the MACS record in the control file and searches for update level identifiers from the top of the control file down. If HCPLDR finds an

update level identifier other than TEXT, the command appends the identifier to TXT to create a file type, then searches for the file (*fn* TXT*xxxxx*). If HCPLDR cannot find an object file that has the first update level identifier, the command finds the next identifier and continues the search. As a last resort, when HCPLDR cannot find the object file according to update level identifiers, the command searches for *fn* TEXT. For example, if the control file is:

```
TEXT MACS HCPM20 HCPM1D
SPEC AUX1111
IBM1 AUXR21
```

HCPLDR follows this search order:

```
fn TXTSPEC
fn TXTIBM1
fn TEXT      (last resort)
```

If an asterisk (*) or a period appears in the first column of a control file record, HCPLDR ignores the record. The command also ignores update level identifiers of PTF.

ft1 [/] [...*ft256*]
is a list of file types (up to 256) that you want HCPLDR to use in lieu of a control file. HCPLDR establishes a search order from left to right. The list of file types can be null, in which case a file type must be specified for each object module in the load list EXEC.

If you place a slash (/) anywhere in the file type list, HCPLDR notifies you when it loads an object module that has a file type to the left of the slash.

You must use the NOCTL option if you specify file types directly on the HCPLDR command line.

Assume you want to establish the following search order for HCPLDR:

```
fn TXT1
fn TXT2
fn TXT3
```

Also, you want to be notified when HCPLDR uses file type TXT1 or TXT2. The load list you are using is CPLOAD EXEC. Enter the following command:

```
hcpldr cpload txt1 txt2 / txt3 (noctl
```

When you enter a list of file types on the command line, HCPLDR does not look for *fn* TEXT automatically.

MAP
requests HCPLDR to produce a load map of the system it creates. The load map is called *loadlist* MAP A1.

This option is not valid if you use the TAPE or the PUNCH option. If you use the TAPE or PUNCH option, the system loader creates a load map when you load (IPL) the tape or reader.

PAGEB
requests that every CSECT be placed on a 4-kilobyte page boundary.

This option is not valid if you use the TAPE or the PUNCH option.

TAPE
sends the output of object files to a tape drive at virtual device number 0182. At the command execution, HCPLDR does not link-edit the system; rather, HCPLDR places a three-card loader and a stand-alone loader before the object modules. When you load (IPL) the tape, the stand-alone loader link-edits the object modules and passes control to the program (HCPGEN) that writes the system onto the system residence device. The stand-alone loader also creates a load map of the system. HCPLDR ignores MAP or PAGEB, if you specify those options along with TAPE.

PUNCH
sends a punch file to virtual punch 000D. At the command execution, HCPLDR does not link-edit the system; rather, the command places a three-card loader and a stand-alone loader before the object modules. You can spool your punch to your virtual reader or some other (System/370 or System/370-XA) virtual machine reader. When loaded (IPL), the stand-alone loader link-edits the object modules and passes control to the program (HCPGEN) that writes the system onto the system residence device. HCPLDR ignores MAP or PAGEB, if you specify those options along with PUNCH.

NOCTL
is the option you must use when you use a list of file types in lieu of a control file. HCPLDR will establish a search order of file types according to what you enter on the command line.

Usage Notes

1. When you use the system loader as a system generator, your virtual machine must have sufficient storage to contain CMS **plus** the entire system you are generating (in the case of CP, this includes any V=R region). If the system is too large to generate directly, use the stand-alone version of the system loader.
2. Using the PAGEB option improves the debugging task since it eliminates needless arithmetic (the lower three hexadecimal digits of the load address is the same as the assemble address). Note, however, that PAGEB should only be used for testing purposes because the option wastes storage.
3. After the new CP nucleus resides on the system residence device, you can shut down your present system (with the SHUTDOWN command) and perform a hardware load of the new system.
4. The following response may appear when you load the CP nucleus:

   ```
   CSECT'S WITH SIZE GREATER THAN CONDITIONAL PAGE
   BOUNDARY @MAPSTRT xxxxxx HCPEMA xxxxxx
   ```

   This response is for information only and is no cause for concern unless other CSECTs are listed. The response indicates that the CSECTs are larger than the specified page size. HCPEMA contains the error message text and @MAPSTRT is the machine-readable load map. CP handles these CSECTs in a special way. If any *other* pageable modules exceed the page size, ensure that you have not added any CPB load control statements to the load file and then notify the IBM Support Center.

Loader Control Statements

The following describe loader control statements, which can be actual cards or card images. The following general rules apply to loader control statements:

1. The first column of the loader control statement must be X'02' (the 12-2-9 punch).
2. Columns 2 through 4 must contain the statement acronym.
3. Unless stated otherwise, operands can appear anywhere between columns 5 and 72.
4. Unless stated otherwise, operands must consist of arguments of no more than eight characters.
5. If a control statement has more than one operand, the operands must be separated by blanks.

Printer Control Statement

The printer control statement enables or disables printing of the cross-reference listings and load maps.

Format:

1	2–4	5–72
X'02'	PRT	{ON / *rdevno* / OFF}

where:

X'02'
must appear in column one.

PRT
is the printer control statement acronym.

{**ON** | *rdevno* }
enables printing of cross-reference listings and the load map. If you specify ON, the printer is at 000E. If you specify rdevno, the printer is at that device number.

OFF
disables printing.

If you issue the HCPLDR command without the PUNCH or TAPE option, this operand is ignored.

Note: Only one operand may appear on the statement.

Set Page Boundary Control Statement

The set page boundary control statement aligns the next CSECT according to the page size specified.

Format:

1	2–4	5–72
X'02'	SPB	*nnnnn*

where:

X'02'
must appear in column 1.

SPB
is the set page boundary control statement acronym.

nnnnn
is the page size in hexadecimal. The page size may be any power of 2 from 8 to 65536. If not specified, the page size is X'1000' (decimal 4096).

Note: Only one argument may appear on the statement.

Unconditional Page Boundary Control Statement

The unconditional page boundary control statement aligns all subsequent CSECTs according to the page size specified.

Format:

1	2–4	5–72
X'02'	UPB	*nnnnn*

where:

X'02'
must appear in column 1.

UPB
is the unconditional page boundary control statement acronym.

nnnnn
is the page size in hexadecimal. The page size may be any power of 2 from 8 to 65536. The initial value is 8. If not specified, the page size is X'1000' (decimal 4096).

Note: Only one argument may appear on the statement.

Conditional Page Boundary Control Statement

The conditional page boundary control statement conditionally aligns all subsequent CSECTs according to the page size specified. If a CSECT will not fit within the remainder of the present page, the system loader will locate the CSECT on the next page boundary.

Format:

1	2–4	5–72
X'02'	CPB	*nnnnn*

where:

X'02'
must appear in column 1.

CPB
is the conditional page boundary control statement acronym.

nnnnn
is the page size in hexadecimal. The page size may be any power of 2 from 8 to 65536. If not specified, the page size is X'1000' (decimal 4096).

Note: Only one argument may appear on the statement.

Usage Note: The standard value for page sizes is X'1000' (decimal 4096). Module HCPMM4 uses this standard size for the CP nucleus.

Set Location Counter Control Statement

The set location counter control statement sets the next CSECT at an absolute address.

Format:

1	2–4	5–23	24–80
X'02'	SLC	*nnnnnnnn*	ignored

where:

X'02'
must appear in column 1.

SLC
is the set location counter acronym.

nnnnnnnn
is the absolute address in hexadecimal.

Note: Only one argument may appear on the statement.

Usage Notes:

1. Module HCPSYS uses the set location counter control statement to set the size of the V=R area.
2. CMS uses the set location counter control statement to set the transient and user areas.

Padding Control Statement

The padding control statement fills all unspecified locations in a CSECT with a specified value.

Format:

1	2–4	5–72
X'02'	PAD	*nn*

where:

X'02'
must appear in column 1.

PAD
is the padding control statement acronym.

nn
is the fill character in hexadecimal. The initial value is 0.

Note: Only one argument may appear on the statement.

Usage Note: Padding helps in debugging a program; if a program references unspecified areas, you can detect such references by padding with a value of X'F0' or some similar value.

Parameter Control Statement

The parameter control statement supplies a parameter string that a loaded system uses upon execution.

Format:

1	2–4	5–72
X'02'	PRM	*parmstring*

where:

X'02'
must appear in column 1.

PRM
is the parameter control statement acronym.

parmstring
is a parameter string having a 2-byte length field followed by the parameter. When the system begins execution, general purpose register one addresses this parameter string.

Subsystem Control Statement

The subsystem control statement specifies a subsystem and the location within a virtual machine where the subsystem appears. Such a subsystem is a separately loaded system, having its own relocation, page boundaries, cross reference listings, and load maps. However, the system loader loads the subsystem as a part of the main system. Use such a "system loaded within a system" for stand-alone programs. You can load the programs as part of the CP nucleus.

Format:

1	2–4	5–72
X'02'	SYS	*name* [*loc*]

where:

X'02'
must appear in column 1.

SYS
is the subsystem control statement acronym.

name
is the name of the subsystem.

loc
is the location where the subsystem appears in the virtual machine. If omitted, the location is 0.

Loader Termination Control Statement

The loader termination control statement terminates the current system or subsystem. Control is passed to the entry point specified on the loader termination control statement. Every subsystem control statement must have its corresponding loader termination control statement and there must be one loader termination statement for the base system.

Format:

1	2–4	5–72
X'02'	LDT	*name*

where:

X'02'
must appear in column 1.

LDT
is the loader termination control statement acronym.

name
is the entry point name that receives control after the system nucleus is loaded into real storage. This operand is ignored for subsystems.

Replace Control Statement

The replace control statement replaces instructions and constants in virtual storage.

Format:

1	2–4	5–6	7–12	13–14	15–16	17–70	71–72	73–80
X'02'	REP		*sadd*		*esdid*	*flds*		nu

where:

X'02'
must appear in column 1.

REP
is the replace control statement acronym.

sadd
is the hexadecimal starting address of the area that will be replaced.

esdid
is the External Symbol Identification, a number assigned by the compiler or assembler to the CSECT in which the replacement occurs.

flds
can be up to eleven 4-digit hexadecimal fields, separated by commas, each replacing one halfword. A comma cannot follow the last field.

nu
means "not used" by the loader. The field may be left blank or contain program identification.

Usage Notes:

1. The replace control statement must be in hexadecimal code.
2. The data in columns 17-70 (excluding commas) on the replace statement replaces what already was loaded into virtual storage beginning at the address specified in columns 7-12.
3. A replace statement may appear:
   - Immediately preceding the END statement of an object module, if the module does not contain relocatable data (such as address constants).
   - Immediately preceding the first relocatable dictionary statement, if the object module does contain relocatable data.
4. If additions made by the replace statements increase the length of a CSECT, an include control section control statement must appear preceding the object module.
5. If the most recent VER control statement for a CSECT failed, the replace operation for that CSECT does not occur.

Verify Control Statement

The verify control statement verifies a corresponding replace control statement, i.e., verify assures that the proper data is replaced.

Format:

1	2–4	5–6	7–12	13–14	15–16	17–70	71–72	73–80
X'02'	VER		*sadd*		*esdid*	*flds*		nu

where:

X'02'
must appear in column 1.

VER
is the verify control statement acronym.

sadd
is the hexadecimal starting address of the area to be verified.

esdid
is the External Symbol Identification, a number assigned by the compiler or assembler to the CSECT in which the verification occurs.

flds
is up to eleven 4-digit hexadecimal fields, separated by commas, each verifying previously loaded halfwords. A comma cannot follow the last field.

nu
means "not used" by the loader. This field may be blank or contain program identification.

Usage Notes:

1. The verify control statement must be in hexadecimal code.
2. The data in columns 17-70 (excluding commas) on the verify statement verifies what already was loaded into virtual storage beginning at the address specified in columns 7-12.
3. Place verify control statements immediately before corresponding replace control statements.
4. If the verify function fails:
   - The loaded system does not execute.
   - All following REP control statements for that CSECT are ignored until the next successful VER control statement for that CSECT.

Include Control Section Control Statement

The include control section statement changes the length of a CSECT. Use it only when replace control statements cause a control section to increase in length.

Format:

1	2–4	5–72
X'02'	ICS	*name size*

where:

X'02'
must appear in column 1.

ICS
is the include control section statement acronym.

name
is the name of the CSECT you wish to enlarge.

size
is the total size required for the CSECT. If size is smaller than the assembled size, the assembled size is used.

Usage Notes:

1. Both arguments in the include control section statement are required.
2. The include control section statement must appear before the first ESD statement of the CSECT.

Delete Control Statement

The delete control statement deletes up to eight CSECTs.

Format:

1	2–4	5–72
X'02'	DEL	*name1* [... *name8*]

where:

X'02'
must appear in column 1.

DEL
is the delete control section statement acronym.

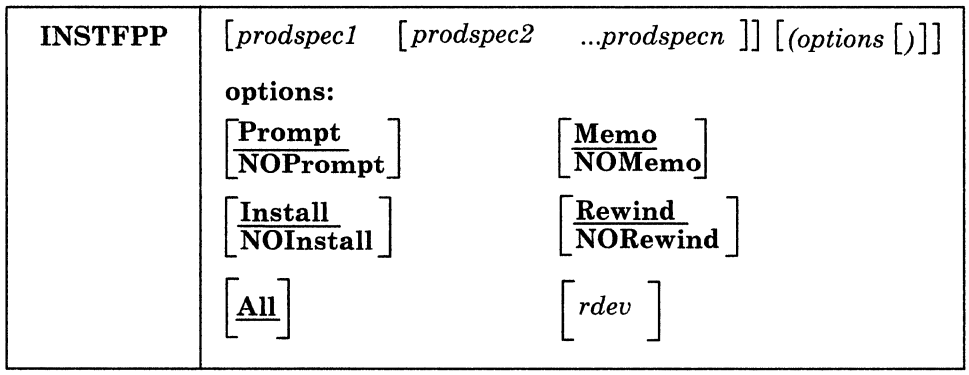
name1 [...*name8*]
is up to eight CSECT names. These names must not be previously defined within the load. Subsequent attempts to define these names as part of the system or subsystem will result in those CSECTs being deleted.

INSTFPP EXEC

Use the INSTFPP EXEC to install licensed programs.

Format

The format of the INSTFPP EXEC is:



prodspec1 ... prodspecn
are the product specification codes that let you specify the products you want processed. These codes consist of the product number and the feature identification code as listed in the FEATURE$ PRODUCTS file. If no feature identification code exists for a product, specify just the product number. If one exists, attach it to the end of the product number. Specify these codes without imbedded hyphens or other punctuation. INSTFPP scans the files on the stacked tape and processes the selected licensed programs.

Prompt
displays the prompts that ask if you want to process the specified licensed programs. PROMPT is the default.

NOPrompt
eliminates the prompts that asks if you want to install the specified licensed programs.

Memo
prints a product Memo to Users from the tape for each selected product. MEMO is the default.

NOMemo
lets you process the selected products without printing product Memo to Users. You cannot specify NOMEMO if you have specified NOINSTALL.

Install
lets you install the selected products. Install is the default.

NOInstall
lets you process the selected products without installing them. You cannot specify NOINSTALL if you have specified NOMEMO.

Rewind
makes sure the tape is rewound before and after product installation and that it is properly mounted. REWIND is the default.

NORewind
lets INSTFPP processing continue without tape rewinds before and after product installation. You can specify NOREWIND only if you invoke INSTFPP from the command line.

Note: Make sure the tape is properly mounted and attached as 181. You can only install products located after the initial tape position.

All
lets you process all the products on the tape. ALL is the default if you do not enter product specification codes.

rdev
lets you specify the real tape address if the tape is mounted on a tape drive but not attached to MAINT as 181.

Before Running INSTFPP

Before you invoke INSTFPP to install licensed programs:

1. Log on to the MAINT user ID.
2. Make sure that MAINT has all privilege classes. Enter:

   ```
   access 291 c ■
   xedit user direct ■
   locate/USER MAINT ■
   ```

   You should see a line that looks something like this:

   ```
   USER MAINT NOLOG 16M 32M ABCDEFG
   ```

   The last field in this line defines the privilege classes assigned to MAINT. If your installation uses the IBM-supplied privilege classes, you should have classes ABCDEFG. If you do not, type in any classes MAINT needs.
3. While you are still looking at MAINT's directory entry, check to see if the MAINT 319 minidisk is defined. If it is not, you must define it.
   a. File your directory.

   ```
   file ■
   ```

   b. Map your USER DIRECT and examine the map to find a space for the 319 minidisk. Look for a gap of appropriate size and record the starting cylinder of that gap. The size you need depends on the number of program products you want to install.

   ```
   diskmap user direct ■
   File USER DISKMAP A has been created.
   ```

   c. XEDIT your directory and add an entry for the new minidisk:

   ```
   xedit user direct ■
   ```

   ```
   locate/USER MAINT ■
   locate/MDISK ■
   ```

   Locate the MAINT minidisk definitions.

   MDISK 319 *devtype startcyl cyl label* **MW ALL** ■

   devtype and *label* are the device type and label of the DASD volume where the new minidisk is located. *startcyl* is the starting cylinder of the gap where the new minidisk is defined. *cyl* is the size of the new minidisk in cylinders.

■ Press **ENTER** to return to the command line.
file ■

4. If you changed the directory, either to authorize new privilege classes for MAINT or to define the MAINT 319 minidisk, issue DIRECTXA to update the directory and bring it online:

```
directxa user ■
VM/XA SYSTEM PRODUCT USER DIRECTORY CREATION PROGRAM - RELEASE 2.0
HCPDIR750W RESTRICTED PASSWORD FILE NOT FOUND
EOJ DIRECTORY UPDATED AND ONLINE
Ready; T=n.nn/n.nn hh:mm:ss
```

5. Link to the MAINT 319 minidisk in read/write mode:

```
link * 319 319 w ■
```

6. Format the 319 minidisk if you have just defined it:

```
format 319 p (blksize 2048 ■
DMSFOR603R FORMAT will erase all the files on disk P(319).
            Do you wish to continue?
            Enter 1 (YES) or 0 (NO).
1 ■
DMSFOR605R Enter disk label:
mnt319 ■
DMSFOR733I Formatting disk P
DMSFOR732I ?? cylinders formatted on P(319)
Ready; T=n.nn/n.nn hh:mm:ss
```

7. Access the 319 minidisk as P:

```
access 319 p ■
```

8. Make sure that you are linked to the following minidisks:

   MAINT 190 (System disk)
   MAINT 191 (MAINT's work disk)
   MAINT 193 (CMS BASE)
   MAINT 194 (CP BASE)

   The INSTFPP EXEC expects these disks to be accessed with these numbers. If your corresponding disks have different numbers, redefine and reaccess them. For example, if your system disk is 490 instead of 190, issue:

```
define 490 190 ■
```

access 190 *fm* ■ *fm* is any unused file mode. You cannot access the redefined system disk as S.

9. Make sure that the INSTFPP code is on one of these disks. Enter:

```
listfile instfpp exec * ■
INSTFPP EXEC S2
Ready; T=n.nn/n.nn hh:mm:ss
```

10. Mount the optional feature product tape. INSTFPP stops if the tape is not mounted correctly.

11. Print a hard copy of your directory:

```
print user direct ■
```

Many product installation EXECs link to the user minidisks in write mode. If the link attempt fails, you may be asked to enter the write password of the minidisk.

12. Verify that you have temporary disk space available on a DASD volume of the same type as your system residence volume. You need at least 30 contiguous cylinders of 3380 temporary disk space or an equivalent amount of 3350 or 3375 disk space. Enter:

```
#cp query alloc
```

```
DASD rdevno volid type TDISK TOTAL=nnnn INUSE=nnnn AVAIL=nnnn
                      PAGE  TOTAL=nnnn INUSE=nnnn AVAIL=nnnn
                      SPOOL TOTAL=nnnn INUSE=nnnn AVAIL=nnnn
                      DRCT  TOTAL=nnnn INUSE=nnnn AVAIL=nnnn [,ACTIVE]
```

The system issues this response for each DASD volume attached to the system. ACTIVE indicates the DASD volume containing the active directory.

If necessary, use the CPFMTXA command to allocate more temporary disk space. See *VM/XA SP CMS Command Reference.*

13. Enter:

```
instfpp (noinstall memo ■
```

to print the product Memo to Users for reference and to copy the Memo to Users to the MAINT 319 minidisk.

14. Make sure that appropriate spooling control options are in effect to direct the virtual printer spool files INSTFPP produces with the PRINT command to the desired real printer. Enter:

```
#cp query tag printer ■
PRT vdevno TAG:
tagtext
```

If you are not satisfied with this setting, change it. You may have to invoke the CP SPOOL command and/or the CP TAG command.

If your printer handles only uppercase characters, use the FOLD option of the CP LOADBUF command. If your printer does not accept the LOADBUF command, print memos by issuing the PRINT command with the UPCASE option. In addition, if your printer cannot print special characters contained in the product memos printed by INSTFPP, look online at the product Memo to Users on the MAINT 319 minidisk.

Refer to *VM/XA SP CP Command Reference* and *VM/XA SP CMS Command Reference* for more information about these commands.

15. Set your virtual storage size to 16MB unless the product Memo to Users specifies otherwise.

```
define storage 16M ■
STORAGE = 16M
STORAGE CLEARED - SYSTEM RESET
ipl cms ■
VM/XA CMS 5.5 mm/dd/yy hh:mm
■
Ready; T=n.nn/n.nn hh:mm:ss
```

16. Mount the product tape on a tape drive and ready the tape drive.

    Warning: Do not attach the tape drive to MAINT. The INSTFPP EXEC will attach it.

Running INSTFPP

You can run INSTFPP in panel mode or by specifying products directly.

- If you want to install several licensed programs, or one licensed program for which you do not know the product specification code, you will find it easier to run INSTFPP in panel mode. Follow the instructions under "Using the INSTFPP Panels."
- If you want to install only a few licensed programs and know the product specification codes, or if you want to install all the licensed programs, you can run INSTFPP either way. If you prefer to specify products directly, go to "Specifying Products Directly" on page 631.

Using the INSTFPP Panels

1. Invoke INSTFPP with no arguments from a 3270 device (20 line minimum).
2. When a panel appears on your screen, enter the real tape drive address, change defaults if necessary, and press **ENTER**.
3. If you choose not to install all products on the product tape, type an X next to the products you want to install.

Figures 38 through 44 on pages 628 through 631 show the INSTFPP panels.

```
SOINS01           INSTFPP INSTALLATION OPTIONS
------------------------------------------------------------------------------

If appropriate, change any defaults and then press the ENTER key.

      Real tape address (will be attached as 181):              ____
      Process all products on the tape (Y/N)?                    Y
      Be prompted before each product is processed (Y/N)?        Y

      Alternatives:
            (1) Install the product(s) and print the memo(s)
            (2) Only print the product memo(s)
            (3) Only install the product(s)
                                        Enter (1, 2, or 3):    1




------------------------------------------------------------------------------
 PF1=Help      2=          3=Quit          4=           5=          6=
 PF7=          8=          9=             10=          11=         12=Cursor

====>
```

Figure 38. INSTFPP Panel 1

Note: If the real tape address appears as R/W, your tape drive is attached to MAINT. Quit and detach the tape drive, then reissue INSTFPP.

```
SOINS02           INSTFPP PRODUCT SELECTION PANEL            Line 1 of 68
------------------------------------------------------------------------------
         Type an X next to the product(s) you want to install
         When you have  finished,  press the PF5 key to begin
         the installation process.

_  5668854          ACF/Network Control Program
_  5664289          ACF/System Support Program
_  5664280          ACF/Virtual Telecommunications Access Method
_  5668899          APL2
_  5668808          Application Prototype Environment
_  5767032          Application System
_  5798RWL          Composition Utility
_  5664329          Contextual File Search/370 for VM/CMS
_  5664296          Cooperative Viewing Facility Version 2
_  5668813          Cross System Product/Application Development
_  5668814          Cross System Product/Application Execution
_  5668918          Cross System Product/Application Query
------------------------------------------------------------------------------
 PF1=Help      2=          3=Quit          4=Return      5=Execute   6=
 PF7=          8=Forward   9=Sort(prodid) 10=Sort(desc) 11=         12=Cursor

====>
```

Figure 39. INSTFPP Panel 2

```
SOINS02            INSTFPP PRODUCT SELECTION PANEL            Line 13 of 68
------------------------------------------------------------------------------
         Type an X next to the product(s) you want to install
         When you have  finished,  press the PF5 key to begin
         the installation process.

_  5748XE4          Directory Maintenance
_  5748XXB          Display Management System for CMS
_  5664370          DisplayWrite/370
_  5748XX9          Document Composition Facility
_  5735XXB          Emulation Program
_  5654260          Environmental Recording Editing and Printing Program
_  5668890          Font Library Service Facility
_  5798DFH          FORTRAN Utilities
_  5668801          Graphical Data Display Manager - IMD
_  5668812          Graphical Data Display Manager - PGF
_  5668812    NL    Graphical Data Display Manager - PGF NL Feature
_  5664200          Graphical Data Display Manager Base
------------------------------------------------------------------------------
 PF1=Help      2=          3=Quit          4=Return      5=Execute   6=
 PF7=Backward  8=Forward   9=Sort(prodid) 10=Sort(desc) 11=         12=Cursor

====>
```

Figure 40. INSTFPP Panel 3

```
SOINS02            INSTFPP PRODUCT SELECTION PANEL            Line 25 of 68
------------------------------------------------------------------------------
         Type an X next to the product(s) you want to install
         When you have  finished,  press the PF5 key to begin
         the installation process.

_  5664200    NL    Graphical Data Display Manager Base NL Feature
_  5668905          Graphical Display And Query Facility
_  5799AXX          Graphics Attachment Support Programming
_  5799BKE          Host Displaywriter Document Interchange
_  5668996          IBM BASIC/VM
_  5664185          IBM High-Accuracy Arithmetic Subroutine Library
_  5798DTE          IBM 3812 Pageprinter VM Support
_  5668897          Info Center/1
_  5668012          Interactive Instructional Presentation System
_  5664282          Interactive System Productivity Facility
_  5664285          Interactive System Productivity Facility/PDF
_  5664204    01    NetView Volume 1
------------------------------------------------------------------------------
 PF1=Help      2=          3=Quit          4=Return      5=Execute   6=
 PF7=Backward  8=Forward   9=Sort(prodid) 10=Sort(desc) 11=         12=Cursor

====>
```

Figure 41. INSTFPP Panel 4

```
SOINS02            INSTFPP PRODUCT SELECTION PANEL            Line 37 of 68
-------------------------------------------------------------------------------
         Type an X next to the product(s) you want to install
         When you have  finished,  press the PF5 key to begin
         the installation process.

_  5664204    02    NetView Volume 2
_  5664204    03    NetView Volume 3
_  5734PL3          OS PL/1 Optimizing Compiler and Libraries
_  5734LM5          OS PL/1 Transient Library
_  5664293          Overlay Generation Language
_  5664199          Page Printer Formatting Aid
_  5796PNQ          Pascal/VS
_  5664298          PC Bond
_  5664312          Print Services Access Facility
_  5664198    B     Print Services Facility/VM Base
_  5664198    S     Print Services Facility/VM 3800
_  5664198    V     Print Services Facility/VM 3820
-------------------------------------------------------------------------------
 PF1=Help      2=          3=Quit          4=Return      5=Execute   6=
 PF7=Backward  8=Forward   9=Sort(prodid) 10=Sort(desc) 11=         12=Cursor

====>
```

Figure 42. INSTFPP Panel 5

```
SOINS02            INSTFPP PRODUCT SELECTION PANEL            Line 49 of 68
-------------------------------------------------------------------------------
         Type an X next to the product(s) you want to install
         When you have  finished,  press the PF5 key to begin
         the installation process.

_  5664309          Professional Office System
_  5664309    PA    Professional Office System Applications Support Feature
_  5668AAA          Query Management Facility
_  5748XP1          RSCS Networking, Version 1
_  5664188          RSCS Networking, Version 2
_  5748XXJ          Structured Query Language/Data System
_  5664191          Virtual Machine Monitor Analysis Program
_  5664319          Virtual Machine/Personal Computer Host Server
_  5746AM2          Virtual Storage Extended/Virtual Storage Access Method
_  5664318          VM/Interactive Productivity Facility
_  5664364          VM Batch Facility
_  5798DMY          VM File Storage Facility
-------------------------------------------------------------------------------
 PF1=Help      2=          3=Quit          4=Return      5=Execute   6=
 PF7=Backward  8=Forward   9=Sort(prodid) 10=Sort(desc) 11=         12=Cursor

====>
```

Figure 43. INSTFPP Panel 6

```
SOINS02            INSTFPP PRODUCT SELECTION PANEL            Line 61 of 68
------------------------------------------------------------------------------
         Type an X next to the product(s) you want to install
         When you have  finished,  press the PF5 key to begin
         the installation process.

_  5748RC1          VM/Pass-through Facility
_  5796PNA          VM Real Time Monitor
_  5664283          VM/Integrated System-Productivity Facility
_  5664291          VMBACKUP Management System
_  5664292          VMTAPE Management System
_  5668958          VS COBOL II
_  5668806          VS FORTRAN
_  5664281          3270 PC File Transfer




------------------------------------------------------------------------------
 PF1=Help      2=          3=Quit          4=Return      5=Execute   6=
 PF7=Backward  8=          9=Sort(prodid) 10=Sort(desc) 11=         12=Cursor

====>
```

Figure 44. INSTFPP Panel 7

Specifying Products Directly

If you enter arguments, the INSTFPP panels do not appear on your screen. You can specify product specification codes by listing them as you find them in the FEATURE$ PRODUCTS file. Omit hyphens or other punctuation marks, and leave one blank between each code. You can specify up to 130 characters, including the command and options, on the CMS command line.

To install all the licensed programs available, specify the ALL option.

After Running INSTFPP

After you run INSTFPP, do the following steps:

1. Execute manual installation and verification procedures as indicated in the product Memo to Users if necessary.
2. Tailor product-dependent files as indicated in the product Memo to Users if necessary.
3. Resave CMS if licensed programs that you installed loaded files to the MAINT 19E minidisk (the Y-disk).
4. Consider placing file directory information for shared, read/only minidisks into a DCSS using the SAVEFD command. Refer to *VM/XA SP CP Command Reference*.

Notes:

1. INSTFPP cannot properly restore minidisks accessed as read/only extensions with a subset defined. INSTFPP reaccesses minidisks as read/only extensions with no subset specification.
2. INSTFPP leaves the tape drive containing the optional feature product tape attached as virtual address 181.
3. A console file is created during the installation process to record terminal activity and is spooled to the MAINT reader when INSTFPP is complete.

PROD LEVEL File

INSTFPP updates a file named "PROD LEVEL" on the MAINT 319 minidisk with the results of each licensed program installation.

Example

Figure 45 shows an example of a PROD LEVEL file.

```
------------------------------------------------------------------------
5748XXB  Display Management System (DMS/CMS)
VER 1    REL 2     MOD 0
Time and date of entry: 20:25:25   30 Sep 1986
*** Product installed and verified successfully
------------------------------------------------------------------------
5799AXX  3277 Graphics Attachment Support PRPQ (GASP)
VER 1    REL 3     MOD 0
Time and date of entry: 21:17:25   30 Sep 1986
*** Product installed; manual verification required
```

Figure 45. Sample PROD LEVEL File

Messages

Each licensed program entry in the PROD LEVEL file has an update message associated with it. The possible update messages and their explanations are:

***** Product installed and verified successfully**

Explanation: The licensed program was installed correctly, and the product was verified successfully.

***** Product files loaded; see the Memo to Users to complete installation**

Explanation: The licensed program files have been loaded successfully. Refer to the product Memo to Users printed by INSTFPP. This memo tells you how to complete the installation of the product and then verify that it installed correctly. In some cases, the product Memo to Users refers to other documentation.

***** Product installed; manual verification required**

Explanation: The licensed program was installed, but it was not verified automatically. Refer to the product Memo to Users printed by INSTFPP. This memo tells you how to make sure the licensed program installed correctly. In some cases, the product Memo to Users refers to other documentation.

***** Product installed; verification failed**

Explanation: The licensed program was installed, but the automatic verification failed. Try to install the licensed program again after correcting any problems; if it does not verify correctly, contact your support personnel.

***** Product Installation EXEC failed; RC =** *rc*

Explanation: The product installation EXEC failed, and the return code passed back by this EXEC to INSTFPP is *rc*. Refer to the product Memo to Users or product installation EXEC prolog to see what this return code means. If you cannot fix the problem, contact your support personnel.

Rerunning INSTFPP

If any licensed programs do not install correctly, take the following steps:

1. Try to solve the problem by using the console log, product Memo to Users, the PROD LEVEL file, and other product-specific documentation.
2. Ready the tape.
3. Invoke INSTFPP.
4. Reinstall products that did not install correctly, plus any products that have these products as prerequisites.

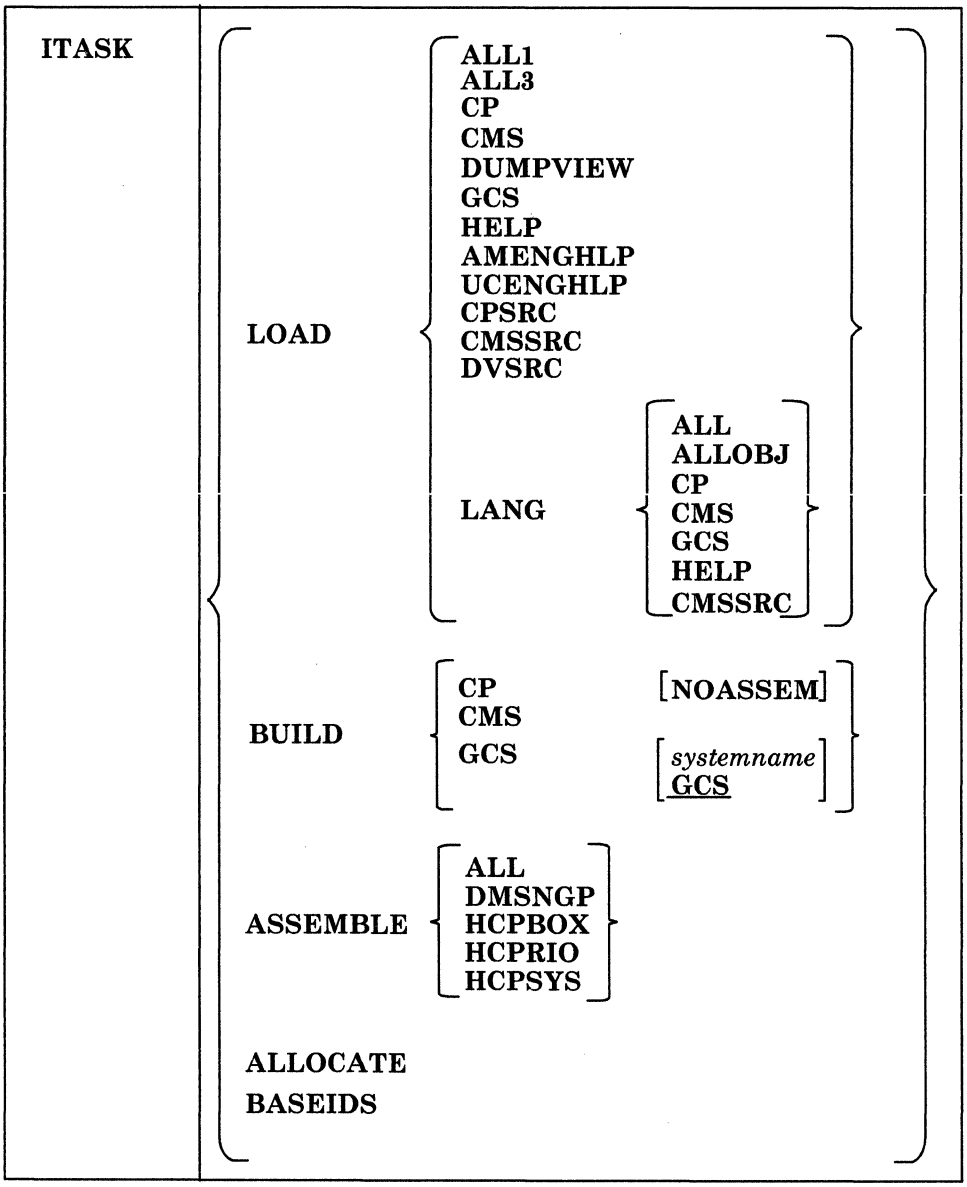
After you install each licensed program, follow the instructions in the product Memo to Users to verify that it has installed correctly (unless it was automatically verified during installation.)

ITASK EXEC

ITASK EXEC is a tool used primarily in the Starter System installation procedure. It invokes other EXECs and commands to perform installation and system generation tasks, to let you complete the installation process with fewer entries and decisions.

Format

The format of the ITASK EXEC is:



LOAD
invokes SPLOAD EXEC to load files from the product tapes, to the minidisks specified in SPLOAD PROFILE. Some LOAD operands may perform additional operations.

Warning: The disk addresses in the SPLOAD PROFILE, user directory, 56643082 $PPF, and SETUP EXEC, and the default addresses used by the ITASK EXEC must all match. If you change an address in any one of these places, you must change it in all the others. We recommend that you do not change any addresses.

ALL1
loads all the CP and dump viewing facility object, delta, and apply files from volume 1 of the product tape.

ALL3
loads all the CP, CMS, and dump viewing facility source files from volume 3 of the product tape.

CP
checks to ensure that the SYSGEN tools are loaded, and then loads the the CP object code, DELTA, and APPLY files from the product tape.

CMS
checks to ensure that the SYSGEN tools are loaded, and then loads the IOCP files, CMS SYSTEM, CMS DELTA, CMS APPLY, and CMS BASE tape files from the product tape; then invokes the ASMGEND EXEC to build the starter system assembler.

DUMPVIEW
loads the dump viewing facility object, DELTA, and APPLY files from the product tape.

GCS
loads the GCS INTERFACE and GCS OBJECT tape files from the product tape.

HELP
loads the HELP FILES tape file from the product tape. First the mixed case American English HELP files are loaded to the 19D disk; then they are copied to the 19C disk and converted to upper case.

AMENGHLP
loads the mixed-case American English HELP files to the 19D disk.

UCENGHLP
loads the mixed-case American English HELP files directly to the 19C disk, then converts them to upper case.

CPSRC
loads the CP SOURCE tape file from the product tape.

CMSSRC
loads the CMS SOURCE tape file from the product tape.

DVSRC
loads the DUMPVIEW SOURCE tape file from the product tape.

LANG
indicates that national language files are to be loaded from a national language feature tape.

ALL
loads the entire contents of the national language feature tape.

ALLOBJ
loads only the object (non-source) code. This consists of the following tape files: CP OBJECT, CMS BASE, and GCS OBJECT.

CP
loads only the CP OBJECT tape files.

CMS
loads only the CMS BASE tape files.

GCS
loads only the GCS OBJECT tape files.

HELP
loads only the HELP FILES tape file.

CMSSRC
loads only the CMS SOURCE tape file.

BUILD
invokes VMFBLD EXEC to build the specified nucleus.

CP
assembles the CP sample files; invokes the VMFBLD EXEC to build the CP nucleus.

If the NOASSEM option is used, the CP sample files are not assembled. This option assumes that these files are already assembled individually.

CMS
invokes the VMFBLD EXEC to build the CMS nucleus.

GCS
modifies a copy of the GCS loadlist (GCSLOAD EXEC) and changes the default configuration file entry (GCS) to match the file name of the GCS configuration file, *systemname* GROUP (if you do not specify a file name, the default is GCS); renames the file type of the configuration file from GROUP to ASSEMBLE; assembles the configuration file; and invokes the VMFBLD EXEC procedure to build and save the GCS nucleus.

ASSEMBLE
assembles the specified sample file:

ALL
assembles the HCPBOX, HCPRIO, and HCPSYS files (but not the DMSNGP file).

DMSNGP
assembles only the DMSNGP file.

HCPBOX
assembles only the HCPBOX file.

HCPRIO
assembles only the HCPRIO file.

HCPSYS
assembles only the HCPSYS file.

ALLOCATE
loads in sample directory from the product tape, prompts for a new password to be used by both MAINT and Operator, replaces the starter system directory with this sample directory, and then loads the following files from volume 1 of the product tape.

File	Contents
SYSGEN TOOLS	$DASD$ CONSTS, DISKMAP EXEC
SYSTEM SAMPLES	DVM PROFILE, SAMPNSS EXEC, starter IOCP, FILECONV SAMPEXEC, CONVSYS SAMPCMDS, CONVUCR SAMPCMDS, COMPSCAN SAMPEXEC, COMPSCAN SAMPLIST, VMFUECP EXEC, VMFUECMS EXEC, VMFUEDV EXEC

CPLOC SAMPLES	HCPBOX ASSEMBLE, HCPRIOXA ASSEMBLE, 33*nn* DIRECT, 33*nn* DISKMAP, HCPSYS*nn* ASSEMBLE
CMSLOC SAMPLES	DMSNGP ASSEMBLE, DMSNGP TEXT, DMSNGP TXTUCENG

After loading the files, the ITASK EXEC renames the following files:

Preload Name	**Name After Load**
33*nn* DIRECT	USER DIRECT
33*nn* DISKMAP	USER DISKMAP
HCPSYS*xx* ASSEMBLE	HCPSYS ASSEMBLE

The placement of these files is determined by the SPLOAD PROFILE.

The ITASK EXEC also allocates space on the DASD volumes identified in the $ALLOC$ user ID in the CP directory.

You are prompted for the real address of each DASD volume, and you have the option to SKIP any volume that you do not want to allocate. Allocation is done according to the entries in the directory; all space not specifically allocated is allocated as PERM.

BASEIDS
issues the CMS FORMAT command to format the remaining minidisks defined in the base CP directory which did not have code loaded to them during the installation process.

The minidisks formatted during this procedure are:

- AUTOLOG 191
- CMSBATCH 195
- DISKACNT 191
- EREP 191
- MAINT or XAMAINT 192, 194, 19C, 19D, 19E, 201, 291, 293, 294, 36E (PVM), 391, 392, 393, 394, 423, 490, 491, 492, 495, 49C, 49D, 501, 591, 592, 593, 594, 595, 596, 59E, 691, 692, 791, 892, 895, 896, 89E

  Note: If you issue ITASK BASEIDS from a user ID other than MAINT or XAMAINT, that user ID's minidisks are formatted instead.
- OPERATNS 191
- RSCS 191.

Also, a PROFILE EXEC is placed on the EREP 191 and DISKACNT 191 minidisks during this procedure.

Messages and Return Codes

DMSWTK002E **File** *fn ft* [*fm*] **not found**

Return Code: 28

DMSWTK003E **Invalid option:** *option*

Return Code: 24

DMSWTK008E **Device** *vdev* **invalid or nonexistent**

Return Code: 36

DMSWTK050E **Parameter missing after** *function*

Return Code: 24

DMSWTK070E **Invalid parameter** *parameter*

Return Code: 24

DMSWTK095E **Invalid address** *vstor*

Return Code: 100

DMSWTK360E **Invalid response** *response*

Return Code: 0

DMSWTK961E **There are class** *class* **files in the** *device*
DMSWTK965I **You may wish to tailor the following files before nuclei generation:**

HCPBOX ASSEMBLE
HCPRIO ASSEMBLE
HCPSYS ASSEMBLE
DMSNGP ASSEMBLE
USER DIRECT

Return Code: 20

DMSWTK966R **Do you wish to have the HELP files converted to uppercase?**

Return Code: 0

DMSWTK967R **Type: (Yes) or No**

Return Code: 0

DMSWTK968I **The following minidisks defined in the base CP directory will be formatted:**

userid	**192**	*userid*	**49C**
userid	**194**	*userid*	**49D**
userid	**19C**	*userid*	**501**
userid	**19D**	*userid*	**591**
userid	**19E**	*userid*	**592**
userid	**201**	*userid*	**593**
userid	**291**	*userid*	**594**
userid	**293**	*userid*	**595**
userid	**294**	*userid*	**596**
userid	**391**	*userid*	**59E**
userid	**392**	*userid*	**691**
userid	**393**	*userid*	**692**
userid	**394**	*userid*	**791**
userid	**423**	*userid*	**892**
userid	**490**	*userid*	**895**
userid	**491**	*userid*	**896**
userid	**492**	*userid*	**89E**
userid	**495**		

Return Code: 0

DMSWTK969I **A PROFILE EXEC has successfully been copied to {EREP's|DISKACNT's} 191 minidisk.**

Return Code: 0

DMSWTK970I **Formatting** *user id vdevno* **minidisk**

Return Code: 0

DMSWTK981R **What is the real address of your** *volume* **volume?**

Return Code: 0

DMSWTK982R **Type: Real address or SKIP**

Return Code: 0

DMSWTK983E **Violation of CMS naming convention found in** *args*

Return Code: 24

DMSWTK1310I **Do you want to format minidisks: AUTOLOG1 191, CMSBATCH 195, DISKACNT 191,** *user id* **36E (PVM), EPEP 191, OPERATNS 191, and RSCS 191?**

Return Code: 0

DMSWTK8005R **Type: 1 (Yes) or 0 (No).**

For a complete explanation of each message, refer to *VM/XA SP System Messages and Codes Reference.*

The Patch Facility

The patch facility is used to make changes to TEXT files in the CP or CMS nucleus.

The patch facility lets you maintain (patch) object code when neither source nor object deck replacements are available. These patches are a temporary solution to object code problems until you receive a replacement file from IBM.

The procedure for maintaining object code is similar to that for maintaining source code. However, with object code, rather than changing source and reassembling the source file, you'll apply changes directly to the TEXT files as the nucleus is generated.

Controlling Patches

You control patches with three kinds of files:

- Control files
- AUXiliary files
- Update files.

Note: As with source updates to base ASSEMBLE files, patches to TEXT files do not cause any changes to the original TEXT files because temporary TEXT files are created. The temporary TEXT files are used to generate the nucleus, then erased.

Here's a description of the files and elements you'll need to understand to make object code patches.

Control File This file contains a MACLIB statement and a list of file types, one for each level of AUX file. It also contains the text deck qualifier for each level. A keyword, TX$, must follow the last AUX file name on a line to flag the AUX file or update as containing text patches.

Note: The patch facility creates a temporary control file with a file name of $$$TUP$$ CNTRL. The use of this name for any control file is restricted to the patch facility.

AUX File This file contains a list of the file types of update files. The update files contain source updates to source or updates to TEXT files.

Note: Any entry in the AUX file which contains the name of a patch update file must contain the TX$ control word between columns 8 and 13.

Load Map This function contains information from both the map of CSECT external symbol resolution and service level information. The map also includes local patches and co-requisite and prerequisite information. The date and time listed in the load map for each patched text file are the original date and time, not the date and time when the temporary deck was created.

TEXT File This file contains the data elements that have been assembled by the customer for source maintenance or provided (already assembled) by IBM for object maintenance. You use it to create executable modules. These files contain APAR corequisite and prerequisite information.

Note: The patch facility creates temporary text files with a file type of TXTTUP. Only the HCPLDR EXEC (the loader EXEC) may use this file type.

Example of a Patch Update File

The following are the functionally equivalent HCPLDR EXEC control statements contained in the update file, which will appear as comments to the update program:

```
./ * * CO-REQ: nnnnnnnn
./ * * PREREQ: nnnnnnnn
./ * NAME CSECTname
./ * *
./ * VER disp data
./ * REP disp data
./ * ICS name size
```

These statements are equivalent to the loader control statements in content. HCPLDR EXEC reformats the statements to make them acceptable to HCPLDR module.

Note: Pre-allocated patch areas are not required because the loader can expand any CSECT in the executable module as it is created from the text files. Addressability to the expanded area is available in every CSECT although space for a patch area need not be. This results in reduced size of executable modules because patch areas do not consume space until they are used.

Here's an explanation of each statement:

Note: Each statement below must be preceded by "./" .

*** NAME** *CSECTname*

This statement identifies which CSECT in a text file is to be patched. It is optional. The *CSECTname* must match the SD name from the ESD for the module. If you omit this statement, the CSECT with the same name as the file name is patched.

*** *** *comment*

This is a comment to the loader EXEC and is optional.

*** * CO-REQ:** *nnnnnnnn*

This is an APAR requisite comment. Use it to indicate dependencies on APARs or other patches. The following format is suggested:

```
./ * * CO-REQ: VM23418
./ * * PREREQ: VM23418
```

When there are no dependencies, put "NONE" after the keywords CO-REQ: and PREREQ:.

*** VER** *disp data*

This statement verifies that the patch is applied at the correct point in the executable module. You need at least one verify statement for each patch. As much data as is required to assure uniqueness should be verified. For example, you may include a verify of the date in the copyright constant of the module prologue.

The *disp* is the displacement, in hex, of the location to be patched in the CSECT.

The *data* is the existing old data in hex to be replaced. This may be up to eleven four digit fields separated by commas. A comma may not follow the last halfword.

*** REP** *disp data*

This statement contains the data to replaced in the CSECT.

The *disp* contains the displacement in hex of the location to be patched in the CSECT.

The *data* is the new data in hex that will replace the existing data. This may be up to eleven four-digit fields separated by commas. A comma may not follow the last half-word.

*** ICS** *name size*

This statement expands the CSECT in the executable module (nucleus). This statement is optional.

The *name* is the same name used in the NAME statement. If a NAME statement is not present, *name* must match the file name of the patch file in which it is included.

The *size* is the total size required for the CSECT.

The following are the functions used in the object service process.

- VMFBLD EXEC

  The VMFBLD EXEC invokes the loader EXEC (that is, HCPLDR EXEC).

- HCPLDR EXEC

  This EXEC exploits the Verify and Replace capabilities of the loader module.

  It gathers TEXT files into an executable nucleus based upon control information.

  Note: This function does not change previous HCPLDR command syntax except to include a "NOPATCH" option.

  This EXEC provides the same level of control for local service, when source is not available, as is currently provided for source file updates. This means that the nucleus-generation process automatically locates and tries to reapply a local patch each time a replacement update file arrives from IBM.

  This results in one of two possibilities:

  1. The local fix is applied and you receive replacement update files. You must then evaluate whether:
     - The local fix is obsolete.

       If the replacement update file contains the APAR fix that replaces the local fix, then you can remove the local fix by removing the object update from the AUX file or the control file.
     - The local fix is still needed.

       If the replacement update file does not contain an APAR fix that replaces the local fix, then you will leave the local fix in place.
  2. The local fix is not applied and you receive replacement service files. You must then evaluate whether:
     - The local fix is obsolete.

       If the replacement update file contains an APAR fix that replaces the local fix, you just accept the replacement deck.
     - The replacement update file does not contain an APAR fix equivalent to the local fix.

       If the replacement update file does not contain an APAR fix that replaces the local fix, you may:
       - Choose the APAR fix as more important, and just accept the replacement update file.
       - Fix both problems by accepting the replacement update file and reworking the local fix.

Compatibility with HCPLDR

A temporary composite TEXT file is created by the loader EXEC; the composite TEXT file is a copy of the IBM-supplied TEXT file with the patch update file merged into it. The temporary text deck will have the same date and time as the original. The NAME statement is not included in a composite TEXT file. The NAME statement is used only to identify the correct CSECT. When this statement is not present, the patch update file changes the CSECT with the same name as the TEXT file.

A composite TEXT file may include:

- The patch AUX file entry (included as a comment), placed following existing comments, but before the first ESD statement.
- An APAR requisite comment, placed after the AUX file comment, but before the first ESD statement.
- An optional ICS statement, placed after an APAR requisite statement, but before the first ESD statement.
- Required VER and REP statements, placed after any ICS statement, but before the first RLD statement or before the END statement if there is no RLD statement in the TEXT deck.
- Patch update file comments, included as they are encountered.

For each NAME statement in a patch update file and for each patch update file without a NAME statement, a VERIFY statement must be present or the patch is not accepted by the loader EXEC. The composite TEXT file will have the same file name and its file type will be TXTTUP.

The loader EXEC works the same way as the current HCPLDR command works. The EXEC goes through the load list one module at a time, using the CNTRL file to determine the file type. For each temporary composite TEXT file, the EXEC provides a file name with the file type TXTTUP. In addition, the EXEC creates a $$$TUP$$ CNTRL file that contains the file type qualifier (TUP that produces a file type of TXTTUP) of the composite TEXT file created by the loader EXEC.

The loader EXEC invokes the HCPLDR MODULE by passing it a temporary control file named $$$TUP$$ along with other parameters. All temporary files such as TXTTUP or $$$TUP$$ have a file mode of 3; these files are erased automatically as soon as they are used.

Usage Notes

The patch facility is provided to give you:

- The same control that you have with source updates and IBM update file updates.
- The same tracking capability so that no previously applied patch will be lost or ignored when IBM replacement service is applied based upon the contents of the load map.

The following guidelines are recommendations to follow:

1. Keep each fix to a TEXT file in a separate update file.

   This also applies to source update fixes; each source update fix should be in a separate update file.

   Each fix should have an alphanumeric number that is the file type of the update file.

2. Keep all local fix descriptions for the same TEXT file in the same AUX file, unless a fix applies to a different control file level.

   Local fixes for the same TEXT file should not be distributed over AUX files (different control file levels) arbitrarily. Local service should be easily distinguished from IBM service and should always be applied last. Local service can be distributed over separate control files for the purpose of maintaining different service levels with a single structure of AUX and update files. Each level can be built from a different control file containing only the desired level identifiers.

3. Never place local patches in AUX files from IBM. In other words, keep your local service separate from IBM service. Local service should be easily distinguished from IBM service and should always be applied last.

4. Patches to TEXT files should be applied only when no source file is available. Mixing source updates and TEXT file patches for the same module will lead to confusion and is not recommended.

Example of Local Service to TEXT Files

The following is an example of a control structure containing patches at more than one level. This is an appropriate use of the patch facility.

File name	File type	Contents
PRODSYS	CNTRL	L3 AUXLCL3 TX$ P1 AUXP1 TEXT AUXXA
CMSSYS	CNTRL	L2 AUXLCL2 TX$ L3 AUXLCL3 TX$ P2 AUXP2 P1 AUXP1 TEXT AUXXA
TESTSYS	CNTRL	L1 AUXLCL1 TX$ L2 AUXLCL2 TX$ L3 AUXLCL3 TX$ P3 AUXP3 P2 AUXP2 P1 AUXP1 TEXT AUXXA
HCPXYZ	AUXLCL1	PATCH3 TX$ APAR3
HCPXYZ	PATCH3	./ * VER 24 4780,C204 ./ * REP 24 4700
HCPXYZ	AUXLCL2	PATCH2 TX$ APAR2
HCPXYZ	PATCH2	./ * VER 254 47F0,6062 ./ * REP 254 4700 ./ * VER 260 5810,7042,5010 ./ * REP 260 58F0,7042,50F0
HCPXYZ	AUXLCL3	PATCH1 TX$ APAR1
HCPXYZ	PATCH1	./ * VER 254 4740,6062 ./ * REP 254 47F0
HCPXYZ	TEXT	ESD TXT RLD END

Example of Local Service to ASSEMBLE Files

The following is an example of a control structure containing a temporary patch over a local source update. Although this works, it is not recommended. Whenever source code is available, you should use source updates instead of patches.

File name	File type	Contents
XASYS	CNTRL	L1 AUXLCL1 TX$ P1 AUXP1 TEXT AUXXA
HCPXYZ	AUXLCL1	TEMP02 TX$ PROB1 LCFIX5 TX$ APAR2
HCPXYZ	TEMP02	./ * VER 24 4780,C204 ./ * REP 24 4700
HCPXYZ	LCFIX5	./ ADD 12340000 XYZLOOP TM FLAG,X'01' CHECK COMPLETE BO DONE EXIT LOOP
HCPXYZ	TXTL1	ESD TXT RLD END

PRELOAD MODULE

The preloader is a utility program that runs under CMS. It collects multiple text files and reformats them into a single text file. The function of the preloader is similar to that of a linkage editor, but the output is in standard text file format and does not include multiple CSECTS.

A program can be developed using separate assembly modules that reference each other. The preloader can then be used to combine the assembled text files into a single loadable text file.

Format

The format of the PRELOAD command is:

PRELOAD	*loadlist* [*control*]

where:

loadlist
specifies the file name of an EXEC on the caller's A-disk or read-only extension containing records that define input to the preloader. Each of these records contains the file name and optionally the file type of an input text file. The format of each loadlist record defining an input file must be:

```
&1 &2 filename  filetype
```

control
optionally specifies the file name of a CNTRL file residing on one of the caller's accessed disks. The format and interpretation of the CNTRL file is the same as that for the VM/SP VMFLOAD utility. It normally contains file types in priority sequence to be used for selecting input files if file types are not included in the loadlist file. The IBM-supplied control files are described in Chapter 7, "How VM/XA System Product Uses Control Files and Update Files" on page 377.

Note: PRELOAD ignores records that have a PTF update level identifier. It then searches for the next lower level identifier to determine the file type of the input text file. PRELOAD also ignores any options in the loadlist.

Input

The preloader gets input file names from the loadlist. The file type for each input file is determined in one of three ways:

1. If the loadlist record for a given input file includes a file type entry, that file type is used to locate the record.
2. If the loadlist record does not contain a file type, and a 'control' parameter was specified on the PRELOAD command line, the file type constructed is in the format TXT*xxxxx*. In this case *xxxxx* is the highest control level identifier in the control file for which a file can be located on the caller's accessed disks.
3. If no file type is specified on the loadlist entry, and a control file has not been specified on the PRELOAD command line, then the default file type value is TEXT.

Note: Input files are located by scanning the caller's disks in their access order. All input files must be on accessed disks.

Output

The preloader output consists of two files written to the caller's A-disk:

1. *fn* TEXT
2. *fn* MAP.

The file name for each of these files is the same as that specified for the input loadlist file. If either of these files already exists on the caller's A-disk, the new copy replaces the old one.

TEXT File

The output TEXT file is a merged and linked composite of the input files. The first CSECT or private code section in the input expands to contain all input files. Its length is the sum of the lengths of the input files, rounded up to doubleword multiples between sections. Input TXT records of non-zero length are relocated and written to the output file.

The output RLD is a translated and relocated collection of all input RLD records. No sorting is done by the preloader. In general, each output ESD, TXT, and RLD entry appears in the same order as the corresponding input entry. ADCON and VCON fields are relocated within their TXT records. ORG statements that cause relocatable constant fields to overlay or be overlaid may cause results that differ from results obtained with a loader that completes TXT data loading prior to relocating ADCONs and VCONS.

MAP File

The output MAP file is a printable record of preloader processing, similar to a load map. The first line of the map contains:

- Output text file name
- Residence volume label and volume device address
- Date and time of file creation.

The next section of the map is a listing of the control file (if any) used. The remainder of the map contains, in processing order, a section for each input file. Each of these sections consists of:

- File name, file type, file mode of input file
- Residence volume label and virtual device address
- Input file's creation date and time
- Any invalid input records.

Messages

PRELOAD issues the following CMS messages:

DMSPRE001E No file name specified
DMSPRE002E File *fn ft fm* **not found**
DMSPRE104S Error *nn* **reading file** *fn ft fm* **from disk**
DMSPRE105S Error *nn* **writing file** *fn ft fm* **on disk**
DMSPRE109S Virtual storage capacity exceeded
DMSPRE183E Invalid {CONTROL|AUX} file control card
DMSPRE234E Error in LOAD LIST file *fn ft fm*[**; no input**]
DMSPRE235E Error *n* **in input text file** *fn ft* [*fm*]
DMSPRE236E Unresolved external reference(s) encountered
DMSPRE237E Duplicate external symbol(s) encountered
DMSPRE238E Preloader processing error

For a complete explanation of each message, refer to *VM/XA SP System Messages and Codes Reference.*

SAMGEN EXEC

The SAMGEN EXEC creates a segment that contains the simulated VSE modules necessary to support Sequential Access Method (SAM) data management (DTFSD), the ESERV utility program, and Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM).

To install this segment, called CMSBAM, use the SAMGEN EXEC procedure.

Before you invoke SAMGEN EXEC, you need to define a virtual machine large enough to contain CMSBAM. To provide room for the loader tables, the size of the virtual machine should be at least 512K greater than the location at which you intend to save CMSBAM. However, you must not define your storage larger than the address at which you loaded CMSDOS.

SAMGEN EXEC performs the following operations:

- Fetches the simulated VSE phases from the CMSBAM DOSLIB file, which is supplied as part of VM
- Loads the simulated phases at the designated address
- Assigns a storage protection key of X'F'
- Saves the segment.

SAMPNSS EXEC

The SAMPNSS EXEC defines named saved systems.

Format

SAMPNSS	{ CMS CMSXA GCS CMSINST HELP CMSDOS CMSBAM CMSVSAM CMSAMS }

where:

CMS CMSXA
creates named saved systems for CMS and CMSXA at pages E00 – FFF.

GCS
creates a named saved system for GCS at pages 0 – 6 (exclusive) and 400 – 5FF (shared).

CMSINST
creates a saved segment for CMSINST at pages C00 – C4F.

HELP
creates a saved segment for HELP at pages C50 – C9F, in the INSTHELP segment space.

CMSDOS
creates a saved segment for CMSDOS at pages B00 – B0F, in the DOSBAM segment space.

CMSBAM
creates a saved segment for CMSBAM at pages B10 – B3F, in the DOSBAM segment space.

CMSVSAM
creates a saved segment for CMSVSAM at pages BA0 – BFF (shared) and A30 – AFF (exclusive), in the DOSBAM segment space.

CMSAMS
creates a saved segment for CMSAMS at pages B40 – B9F (shared) and A00 – A2F (exclusive), in the DOSBAM segment space.

The SAMPNSS EXEC:

1. Defines the named saved systems or segments that you request
2. Queries CP about the status of the named saved systems or segments
3. Displays the status of the named saved systems or segments as shown below:

```
OWNERID  FILE  TYPE  CL  RECS  DATE    TIME   FILENAME   FILETYPE
CMSMAINT nnnn   NSS  A   nnnn  mm:dd   hh:mm   nss       NSS
```

Messages

For information on error messages and codes, see *VM/XA SP System Messages and Codes Reference.*

HCPNSD440I {**Named Saved System (NSS)|Saved segment**} *systemname* **was successfully defined in fileid** *fileno.*

HCPNSS440I {**Named Saved System (NSS)|Saved segment**} *systemname* **was successfully saved in fileid** *fileno.*
Unrecognized Parameters passed — *parm1 parm2*

Explanation: SAMPNSS detected an error in the specified parameter.

SETUP EXEC

Use the SETUP EXEC to access all the disks where the service EXECs and the product parameter file might be found. The SETUP EXEC does not access the disks needed by the service EXECs; each EXEC does that for itself (usually by invoking VMFSETUP).

The access order is:

Mode	Address	Function
A	191	MAINT's work disk
B	395	CMS alternate LOCAL1 disk
E	691	CMS intermediate alternate LOCAL1 disk
F	391	CMS current LOCAL1 disk
G	692	CMS alternate APPLY disk
H	392	CMS current APPLY disk
I	593	CMS alternate DELTA disk
J	293	CMS current DELTA disk
K	193	CMS current BASE1 disk
L	393	CMS current BASE2 disk
M	490	CMS alternate BUILD1 disk
Note: No C- or D-disk is accessed. If you have a C-disk accessed, release it. You do not need to release your D-disk.		

Format

The format of the SETUP EXEC is:

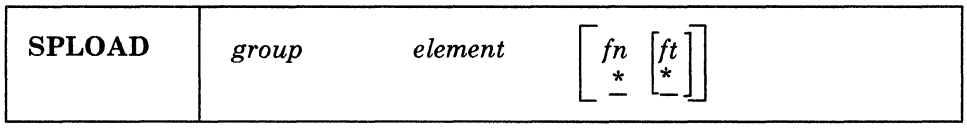
SETUP	

SPLOAD EXEC

The SPLOAD EXEC loads files from the product tapes according to the load instructions contained in SPLOAD PROFILE. SPLOAD PROFILE is described following this section.

Format

The format of the SPLOAD EXEC is:



where:

group element
is a tape file identifier that SPLOAD EXEC uses to locate the entry in SPLOAD PROFILE that contains the load instructions for this tape file.

fn [ft]
form a file specification template. By default, both are set to asterisks (*), meaning all files within a tape file. However, they may be set to a specific file name and/or file type to selectively load specific files from a tape file to a minidisk.

SPLOAD PROFILE

In the sample SPLOAD PROFILE, data entries are organized by format number (see definition below), and format number sets are separated by comment lines. A comment line begins with a slash asterisk (/*). Each data entry in SPLOAD PROFILE has the following syntax:

```
group   element   user id   address   format   volume   fileno
```

where:

group element
is a tape file identifier that identifies the profile entry containing the load instructions for this tape file.

user id
is the owner of the minidisk to which the tape file contents are loaded. This information is used only if SPLOAD EXEC must LINK and ACCESS the minidisk in order to load the file.

address
is the minidisk address to which the tape file is loaded.

format
is the tape format. SPLOAD PROFILE contains a complete load table for each tape format, describing the locations of the files on the product tapes. SPLOAD EXEC selects the appropriate entries in the profile depending on the user's tape format.

volume
is the product tape volume on which the particular tape file resides.

fileno
is the relative tape file number (location) of the specified tape file on the particular product tape volume.

```
!-------------------------------------------------------------!
!              6250 BPI VM/XA Merged Product Tape             !
!-------------------------------------------------------------!
!
! Volume 1
!
Memo      Files      MAINT     191   XASP20V1   1        2
Install   Tools      MAINT     191   XASP20V1   1        3
Sysgen    Tools      MAINT     193   XASP20V1   1        4
System    Samples    MAINT     191   XASP20V1   1        5
CPLOC     Samples    MAINT     295   XASP20V1   1        6
CMSLOC    Samples    MAINT     395   XASP20V1   1        7
CP        Object     MAINT     194   XASP20V1   1        8
CP        Putapply   MAINT     192   XASP20V1   1        9
CP        Putdelta   MAINT     294   XASP20V1   1       10
CP        Corapply   MAINT     291   XASP20V1   1       11
CP        Cordelta   MAINT     291   XASP20V1   1       12
DUMPVIEW  Object     MAINT     193   XASP20V1   1       13
DUMPVIEW  Putapply   MAINT     392   XASP20V1   1       14
DUMPVIEW  Putdelta   MAINT     293   XASP20V1   1       15
DUMPVIEW  Corapply   MAINT     391   XASP20V1   1       16
DUMPVIEW  Cordelta   MAINT     391   XASP20V1   1       17
!
! Volume 2
!
CMS       Base       MAINT     193   XASP20V2   2        2
CMS       Putapply   MAINT     392   XASP20V2   2        3
CMS       Putdelta   MAINT     293   XASP20V2   2        4
CMS       Corapply   MAINT     391   XASP20V2   2        5
CMS       Cordelta   MAINT     391   XASP20V2   2        6
IOCP      Files      MAINT     193   XASP20V2   2        7
CMS       System     MAINT     190   XASP20V2   2        8
GCS       Object     MAINT     595   XASP20V2   2        9
GCS       Interface  MAINT     193   XASP20V2   2       10
GCS       Putapply   MAINT     592   XASP20V2   2       11
GCS       Putdelta   MAINT     596   XASP20V2   2       12
GCS       Corapply   MAINT     491   XASP20V2   2       13
GCS       Cordelta   MAINT     491   XASP20V2   2       14
AMENGHLP  Files      MAINT     19D   XASP20V2   2       15
UCENGHLP  Files      MAINT     19C   XASP20V2   2       15
!
```

Figure 46 (Part 1 of 2). Sample SPLOAD PROFILE

```
! Volume 3
!
CP        Source     MAINT      394   XASP20V3    3         2
CMS       Source     MAINT      393   XASP20V3    3         3
DUMPVIEW  Source     MAINT      393   XASP20V3    3         4
!
!
/*---------------------------------------------------------------------*/
/*          6250 BPI VM/XA NLS Feature Tape                            */
/*---------------------------------------------------------------------*/
/*                                                                     */
CMS       Base       MAINT      193   R2M0NLS     1         2
CP        Object     MAINT      194   R2M0NLS     1         3
HELP      Files      MAINT      ?     R2M0NLS     1         4
CMS       Source     MAINT      393   R2M0NLS     1         5
DUMMY     Object     MAINT      XXX   R2M0NLS     1         6
GCS       Object     MAINT      595   R2M0NLS     1         7
/*                                                                   */
/*                                                                   */
```

Figure 46 (Part 2 of 2). Sample SPLOAD PROFILE

Warning: The disk addresses in the SPLOAD PROFILE, user directory, 56643082 $PPF, and SETUP EXEC, and the default addresses used by the ITASK EXEC must all match. If you change an address in any one of these places, you must change it in all the others. We recommend that you do not change any addresses.

Usage Notes

1. SPLOAD must do a tape rewind each time it is invoked to make sure that the proper product tape is mounted and to be able to position the tape to the proper tape file. If the incorrect tape is mounted then SPLOAD will prompt for the proper product tape.

   The first tape file on each product tape contains a header file named $TAPE$ HEADER. The first line of this file must be in the following format:

   `TAPENO = n ; TAPEFORMAT = f`

   where *n* is the relative tape number and *f* is the tape format (e.g. R1M05500).

2. SPLOAD expects that the tape drive being used is ATTACHed at virtual address 181. If not, SPLOAD will not be able to continue.

3. SPLOAD will ACCESS the target minidisk at an unused mode letter nearest the end of the alphabet. It will use Z if all modes are used.

4. SPLOAD restores the minidisk access order to its original state just before exiting. However, SPLOAD will not be able to recreate mode extension situations.

5. If SPLOAD does a LINK to a minidisk, followed by an ACCESS to that disk and receives a return code of 100 from the ACCESS, an attempt will be made to FORMAT the disk.

   SPLOAD will provide a minidisk label to FORMAT which will consist of the first three characters of the user ID which owns the minidisk, concatenated with a 3-digit minidisk address. If the minidisk address is four digits long, then only the first two characters of the user ID will be used.

   Note: For user ID MAINT, the default characters used will be either MNT or MT, depending on the size of the minidisk address.

Messages and Return Codes

DMSWTK002E *fn ft* **not found**

Return Code: 28

DMSWSL032E **Invalid file type** *ft*

Return Code: 24

DMSWSL050E **Parameter missing after** *value*

Return Code: 24

DMSWSL062E **Invalid character** *char* **in fileid** *fn*

Return Code: 20

DMSWTK070E **Invalid parameter** *parameter*

Return Code: 24

DMSWSG095E **Invalid address** *vaddr*

Return Code: 100

DMSWSL252E **Invalid file name** *file name*

Return Code: 24

DMSWSL409I **Loading** *fn ft* **to the** *vdev* **disk attached to** *user id*

Return Code: 0

DMSWSL737R **Enter the minidisk address for the** *group element*

Return Code: 0

DMSWSL963E *keyword value* **not found in** *fn ft fm*

Return Code: 24

DMSWSL964R **Wrong tape mounted; mount product tape** *n*
Press ENTER when the correct tape is mounted or type QUIT

Return Code: 0

DMSWSL986I **Unable to restore ACCESS to** *mdisk*

Return Code: 0

For a complete explanation of each message, refer to *VM/XA SP System Messages and Codes Reference.*

UPDATE Command

Use the UPDATE command to modify program source files. The UPDATE command accepts a source input file and one or more files containing UPDATE control statements and updated source records; then it creates an updated source output file, an update log file indicating what changes, if any, were made, and an update record file if more than a single update file is applied to the input file.

Format

The format of the UPDATE command is:

```
UPDATE   fn1 [ft1 | ASSEMBLE [fm1 [fn2 [ft2 [fm2]]]] | A1 ]] [(options ... [])]]

         options:

         [REP | NOREP] [SEQ8 | NESEQ8] [INC | NOINC] [CTL | NOCTL] [OUTMODE fm]

         [STK | NOSTK] [TERM | NOTERM] [DISK | PRINT] [STOR | NOSTOR]
```

where:

fn1 ft1 fm1
is the file identifier of the source input file. The file must consist of 80-character card image records with sequence fields in positions 73 through 80 or 76 through 80. If the file type or file mode is omitted, ASSEMBLE and A1 are assumed, respectively.

fn2 ft2 fm2
is the file identifier of the update file. If the NOCTL option is in effect, this file must contain UPDATE control statements and updated source records. The default file identifier is fn1 UPDATE A1. If the CTL option is specified, this file must be a control file that lists the update files to be applied; the default file identifier is *fn1* CNTRL A1.

REP
creates an output source file with the same file name as the input source file. If the output file is placed on the same disk as the input file, the input file is erased.

NOREP
retains the old file in its original form, and assigns a different file name to the new file, consisting of a dollar sign ($) plus the first seven characters of the input filename (fn1).

SEQ8
specifies that the entire sequence field (columns 73 through 80) contains an eight-digit sequence number on every record of source input.

NOSEQ8
specifies that columns 73 – 75 contain a three-character label field, and that the sequence number is a five-digit value in columns 76 – 80.

Note: Source files sequenced by the CMS editor are sequenced, by default, with five-digit sequence numbers.

INC
increases sequence numbers in columns 73 through 80 in each record inserted into the updated output file, according to specifications in UPDATE control statements.

NOINC
puts asterisks (********) in the sequence number field of each updated record inserted from the update file.

CTL
specifies that *fn2*, *ft2*, and *fm2* describe an update control file for applying multiple update files to the source input file.

Note: The CTL option implies the INC option.

NOCTL
specifies that a single update file is to be applied to the source input file.

OUTMODE *fm*
specifies that UPDATE writes the files it creates on the fm disk. UPDATE writes the files as outlined in the note below. If you do not specify a filemode when using OUTMODE, the file mode number defaults to "1". This disk must be an accessed CMS read/write disk or UPDATE terminates.

Note: UPDATE takes the following steps to determine the disk upon which it places output files. The search stops as soon as one of the following steps is successful:

1. If you specify the OUTMODE option, then UPDATE places the output files on the specified disk. That disk must be read/write. If you accessed the disk as a read only extension, then UPDATE displays the following message:

   ```
   DMSUPD037E DISK fm IS READ-ONLY
   ```

2. If the disk on which the original source file resides is read/write, then UPDATE places the output files on that disk.
3. If that disk is a read-only extension of a read/write disk, then UPDATE places the output files on that particular read/write disk.
4. If neither of the other steps is successful, then UPDATE places the output files on the primary read/write disk (the A-disk).

STK
stacks information from the control file in the CMS console stack. STK is valid only if the CTL option is also specified and is useful only when the UPDATE command is executed in an EXEC procedure.

NOSTK
does not stack control file information in the console stack.

TERM
displays warning messages at the terminal whenever a sequence or update control card error is discovered. (Such warning messages appear in the update log, whether they are displayed at the terminal or not.)

NOTERM
suppresses the display of warning messages at the terminal. However, error messages that terminate the entire update procedure are displayed at the terminal.

DISK
places the update log file on disk. This file has a file identifier *fn* UPDLOG, where *fn* is the filename of the file being updated.

PRINT
prints the update log file directly on the virtual printer.

STOR
specifies that the source input file is to be read into storage and the updates performed in storage prior to placing the updated source file on disk. This option is meaningful only when used with the CTL option since the benefit of increased processing speed is realized when processing multiple updates. STOR is the default when CTL is specified.

NOSTOR
specifies that no updating is to take place in storage. NOSTOR is the default when single updates are being applied (CTL is omitted from the command line).

Update Control Statements

The UPDATE control statements let you insert, delete, and replace source records, as well as resequence the output file.

All references to the sequence field of an input record refer to the numeric data in columns 73 – 80 of the source record, or columns 76 – 80 if NOSEQ8 is specified. Leading zeros in sequence fields are not required. If no sequence numbers exist in an input file, a preliminary UPDATE with only the "./ S" control statement can be used to establish file sequencing.

Sequence numbers are checked while updates are being applied; an error condition results if any sequence errors occur in the update control statements, and warnings are issued if an error is detected in the sequencing of the input file. Any source input records with a sequence field of eight blanks are skipped, without any indication of a sequence error. Such records may be replaced or deleted only if they occur within a range of records that are being replaced or deleted entirely and if that range has limits with valid sequence numbers. There is no means provided for specifying a sequence field of blanks on an UPDATE control statement.

Control Statement Formats:

All UPDATE control statements are identified by the characters "./" in columns 1 and 2 of the 80-byte record, followed by one or more blanks and additional, blank-delimited fields. Control statement data must not extend beyond column 50.

SEQUENCE Control Statement

Use the sequence control statement to resequence the updated source output file in columns 73 – 80 (if SEQ8 is specified), or in columns 76 – 80 with the label placed in columns 73 – 75 (if NOSEQ8 is specified).

The format of the SEQUENCE control statement is:

```
./ S [seqstrt [seqincr [label]]]
```

where:

seqstrt
is a one- to eight-digit numeric field specifying the first decimal sequence number to be used. The default value is 1000 if SEQ8 is specified and 10 if NOSEQ8 is specified.

seqincr
is a one- to eight-digit numeric field specifying the decimal increment for resequencing the output file. The default is the "seqstrt" value.

label
is a three-character field to be duplicated in columns 73 – 75 of each source record if NOSEQ8 is specified. The default value is the first three characters of the input filename (*fn1*).

If you use the SEQUENCE statement, it must be the first statement in the update file. If any valid control statement precedes it, the resequence operation is suppressed.

When the sequence control statement is the first statement processed, the sequence numbers in the source file are checked and a warning message is issued for any errors. If the sequence control statement is processed after updates have been applied, no warning messages will be issued.

Each source record, including unchanged records from the source file and records inserted from the update file, is resequenced in columns 73−80 as it is written onto the output file.

INSERT Control Statement

Use the insert control statement to insert all records following it, up to the next control statement, into the output file.

The format of the INSERT control statement is:

```
./  I  seqno  [$  [seqstrt  [seqincr]]]
```

where:

seqno
is the sequence number of the record following which new records are to be added.

$
is an optional delimiter indicating that the inserted records are to be sequenced by increments.

seqstrt
is a one- to eight-digit numeric field specifying the first decimal number to be used for sequencing the inserted records.

seqincr
is a one- to eight-digit numeric field specifying the decimal increment for sequencing the inserted records.

All records following the "./ I" statement, up to the next control statement, are inserted in the output file following the record identified by the *seqno* field. If the NOINC option is specified, each inserted record is identified with asterisks (********) in columns 73−80. If either the INC or CTL option is specified, the records are inserted unchanged in the output file, or they are sequenced according to the *seqstrt* and *seqincr* fields, if the dollar sign ($) key is specified.

The default sequence increment, if the dollar sign is included, is determined by using one tenth of the least significant, nonzero digit in the *seqno* field, with a maximum of 100. The default *seqstrt* is computed as *seqno* plus the default *seqincr*. For example, the control statement:

```
./  I  2600 $ 2610
```

causes the inserted records to be sequenced XXX02610, XXX02620, and so forth (NOSEQ8 assumed here). For the control statement:

```
./  I  240000 $
```

the defaulted *seqincr* is the maximum, 100, and the starting sequence number is 240100. SEQ8 is assumed, so the inserted records are sequenced 00240100, 00240200, and so forth.

If either INC or CTL is specified but the dollar sign is not included, whatever sequence number appears on the inserted records in the update file is included in the output file.

DELETE Control Statement

Use the delete control statement to delete one or more records from the source file.

The format of the DELETE control statement is:

```
./  D  seqno1 [seqno2]  [$]
```

where:

seqno1
is the sequence number identifying the first or only record to be deleted.

seqno2
is the sequence number of the last record to be deleted.

$
is an optional delimiter indicating the end of the control fields.

All records of the input file, beginning at *seqno1*, up to and including the *seqno2* record, are deleted from the output file. If the *seqno2* field is omitted, only a single record is deleted.

REPLACE Control Statement

Use the replace control statement to replace one or more input records with updated records from the update file.

The format of the REPLACE control statement is:

```
./  R seqno1 [seqno2] [$ [seqstrt [seqincr]]]
```

where:

seqno1
is the sequence number of the first input record to be replaced.

seqno2
is the sequence number of the last record to be replaced.

$
is an optional delimiter key indicating that the substituted records are to be sequenced incrementally.

seqstrt
is a one- to eight-digit numeric field specifying the first decimal number to be used for sequencing the substituted records.

seqincr
is a one- to eight-digit numeric field specifying the decimal increment for sequencing the substituted records.

All records of the input file, beginning with the *seqno1* record, up to and including the *seqno2* record, are replaced in the output file. They are replaced with the records that follow the "./ R" statement in the update file, and precede the next control statement.

As with the "./ D" (delete) function, if the *seqno2* field is omitted, only a single record is replaced, but it may be replaced by more than a single inserted record. The "./ R" (replace) function is performed as a delete

followed by an insert: thus, the number of statements inserted need not match the number deleted. The dollar sign ($), *seqstrt*, and *seqincr* processing is identical to that for the insert function.

COMMENT Statement

Use the comment statement to insert comments into the file.

The format of the COMMENT statement is:

```
./ * [comment]
```

where:

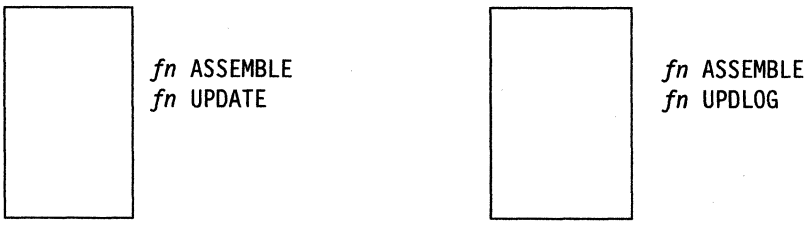
*
indicates that this is a comment statement and is only copied into the update log file.

Summary of Files Used by the UPDATE Command

The following discussion shows input and output files used by the UPDATE command for a:

- Single-level update
- Multilevel update
- Multilevel update with an auxiliary control file.

Single-Level Update



where:

fn **ASSEMBLE**
is the source input file.

fn **UPDATE**
contains UPDATE control statements and updated source input records.

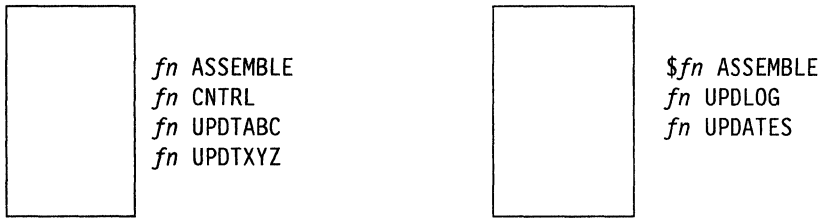
$*fn* **ASSEMBLE**
is the updated source file, incorporating changes, additions, and deletions specified in the update file. The output file type is always the same as the file type of the input file. These default file types and file modes can be overridden on the command line; for example:

```
update testprog cobol b fix cobol b (rep ■
```

results in a source file TESTPROG COBOL B being updated with control statements contained in the file FIX COBOL B. The output file replaces the existing TESTPROG COBOL B.

fn **UPDLOG**
contains a record of updates applied. If you do not want this file written on disk, specify the PRINT option.

Multilevel Update



where:

fn **ASSEMBLE**
is the source input file.

fn **CNTRL**
is the control file that lists updates to be applied to the source file. These default file types and file modes can be overridden on the command line; for example:

```
update acct pliopt a test cntrl a (ctl ■
```

results in the file TEST CNTRL being used by the UPDATE command to locate the update files for ACCT PLIOPT.

fn **UPDTABC**
fn **UPDTXYZ**
are update files containing UPDATE control statements and new source records. These files must have file names that are the same as the source input file. The first four characters of the file type must be "UPDT". The UPDATE command searches all accessed disks to locate the update files.

$*fn* **ASSEMBLE**
is the updated source file, incorporating changes, additions, and deletions specified in the update files. The file type is always the same as the file type of the source input file.

fn **UPDLOG**
contains a record of updates applied. If you do not want this file written on disk, specify the PRINT option.

fn **UPDATES**
summarizes the updates applied to the source file.

The CONTROL FILE (*fn* CNTRL) may not contain UPDATE control statements. It may only list the file types of the files that contain UPDATE control statements. This control file contains the records:

```
TEXT MACS CMSLIB
TWO UPDTABC
ONE UPDTXYZ
```

where UPDTABC and UPDTXYZ are the file types of the update files. The UPDATE command applies these updates to the source file beginning with the last record in the control file. Thus, the updates in *fn* UPDTXYZ are applied before the updates in *fn* UPDTABC.

When you create update files whose file types begin with "UPDT", you may omit these characters when you list the updates in the control file; thus, the CNTRL file may be written:

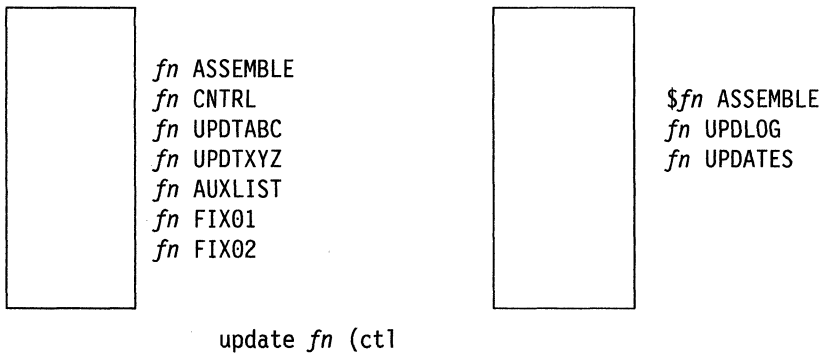
```
TEXT MACS CMSLIB
TWO ABC
ONE XYZ
```

TEXT, TWO, ONE: The first column of the control file consists of an update level identifier, which may be from one to five characters long. These identifiers are used by VM updating procedures, like the VMFHASM EXEC, to locate and identify text decks produced by multilevel updates.

MACS: The first record in the control file must be a MACS record that contains an update level identifier (TEXT) and, optionally, lists up to nine macro library (MACLIB) file names.

UPDATE uses the information provided in the MACS card and the update level identifier only when the STK option is specified. This information is, however, required in the CNTRL file.

Multilevel Update with Auxiliary Control File



***fn* ASSEMBLE, *fn* CNTRL, *fn* UPDTABC, *fn* UPDTXYZ, $*fn* ASSEMBLE, *fn* UPDLOG,** and **fn UPDATES** are used as described, for "Multilevel Update", except that the CNTRL file contains:

```
TEXT MACS CMSLIB
TWO UPDTABC
ONE UPDTXYZ
TEXT AUXLIST
```

AUX in the file type AUXLIST indicates that this is the file type of an auxiliary control file that contains an additional list of updates. The first three characters of the file type of an auxiliary control file must be "AUX"; the remaining character(s) (to a maximum of five) may be anything. The filename must be the same as the source input file.

An auxiliary file may also be specified as:

```
xxxxx AUX
```

in the control file. For example, the record:

```
FIX TEST AUX
```

identifies the auxiliary file *fn* AUXTEST.

Note that if you give an auxiliary control file the file type AUXPTF or an update level identifier of AUX, the UPDATE command assumes that it is a simple update file and does not treat it as an auxiliary file.

Preferred AUX File

A preferred AUX file may be specified. A preferred AUX file contains the version of an update that applies to your version of the source file. (There may be more than one version of the same update if there is more than one version of the source file. For example, you need one version for the source file that has a system extension program product installed, and you need another version for the source file that does not have a program product installed.)

When you specify an auxiliary control file, you can specify more than one file type. The first file type indicates a file that UPDATE uses only on one condition: the files that the second and subsequent file types indicate do not exist. If they do exist, this AUX file entry is ignored and no updating is done. The files that the second and subsequent file types indicate are preferred because, if they exist, UPDATE does not use the file that the first file type indicates. For example, assume that the file *fn* ASSEMBLE does exist. The control file MYMODS CNTRL:

```
TEXT  MACS  MYMACS  CMSLIB  OSMACRO
MY2   AUXTEST
MY1   AUXMINE AUXTEST
```

and the command:

```
update fn assemble * mymods cntrl (ctl ■
```

would result in UPDATE finding the preferred auxiliary control file *fn* AUXTEST, and therefore not using *fn* AUXMINE to update *fn* ASSEMBLE. UPDATE would then proceed to the MY2 AUXTEST entry and update "fn ASSEMBLE" with the updates listed in *fn* AUXTEST. It is assumed that AUXTEST and AUXMINE list similar but mutually exclusive updates.

The search for a "preferred" AUX file will continue until one is found or until the token is an invalid file type; that is, less than four or more than eight characters. This token and the remainder of the line are considered a comment.

fn FIX01 and *fn* FIX02 are update files containing UPDATE control statements and new source records to be incorporated into the input file. When update files are listed in an auxiliary control file, they can have any file type you choose, but the file name must be the same as the source input file. The update files, as well as the AUX file, may be on any accessed disk. These are indicated in *fn* AUXLIST as follows:

```
FIX02
FIX01
```

The updates are applied from the bottom of the auxiliary control file. Thus, *fn* FIX01 is applied to the source file before *fn* FIX02. Since the auxiliary control file is listed at the bottom of the control file, these updates are applied before UPDTXYZ and UPDTABC.

Additional Control File Records

In addition to the MACS record, the file types of update (UPDT) files, and the file types of auxiliary control (AUX) files, a control file may also contain:

- Comments. These records begin with an asterisk (*) in column 1. Comments are also valid in AUX files.
- PTF records. If the characters PTF appear in the update level identifier field, the UPDATE command expects the second field to contain the file type of an update file. The file type may be anything; the file name must be the same as the source input file.
- Update level identifiers not associated with update files.

The following example of a control file shows all the valid types of records:

```
* Example of a control file
ABC MACS MYLIB
TEXT
004  UPDTABC
003  XYZ
002  AUXLIST1
001  LIST2  AUX
PTF  TESTFIX
```

The STK Option

The STK (stack) option is valid only with the CTL option and is meaningful only when the UPDATE command is invoked within an EXEC procedure.

When the STK option is specified, UPDATE stacks the following data lines in the console stack:

```
first line:  * update level identifier
second line: * library list from MACS record
```

The update level identifier is the identifier of the most recent update file that was found and applied. For example, if a control file contains:

```
TEXT MACS CMSLIB OSMACRO TESTMAC
OFA  UPDTOFA
PFA  UPDTOFA
```

and the UPDATE command appears in an EXEC as follows:

```
UPDATE SAMPLE (CTL STK &READ VARS &STAR &TEXT &READ VARS
&STAR &LIB1 &LIB2 &LIB3 &LIB4
```

then the variable symbols set by the &READ VARS statements have the following values if the file SAMPLE UPDTOFA is found and applied to the file SAMPLE ASSEMBLE:

Symbol	*Value*
&STAR	*
&TEXT	OFA
&LIB1	CMSLIB
&LIB2	OSMACRO
&LIB3	TESTMAC
&LIB4	null

The library list may be useful to establish macro libraries in a subsequent GLOBAL command within the EXEC procedure. If no update files are found, UPDATE stacks the update level identifier on the MACS record.

Message

FILE *fn ft fm*, **REC** *#n* **= update control statement**

Explanation: This message is displayed when the TERM option is specified and an error is detected in an update file. It identifies the file and record number where the error is found.

Return Codes

The UPDATE command issues the following return codes:

Return Code	Possible Causes
4	• No file name specified • Input file sequence error
8	• Sequencing overflow • Sequence increment is zero • Sequence error introduced in output file
12	• Premature EOF of file; sequence number not found • Missing PTF file • ./s not first card in input file; card ignored • Invalid char in sequence field • Sequence number not found • Invalid update file control card
24	• Invalid option • Invalid mode • Option specified twice • Conflicting options • Invalid parameter • Option STK invalid without CTL
28	• File not found • File UPDATE CMSUT1 *fm* already exists
32	• File is not fixed, 80-character records • Missing or duplicate MACS card in control file • Invalid file control card
36	• Disk is read-only • Disk not accessed
40	• No update files were found
41	• Insufficient storage to begin update • Insufficient storage to complete update
100	• Error reading file from disk • Error writing file on disk

UTILITY EXEC

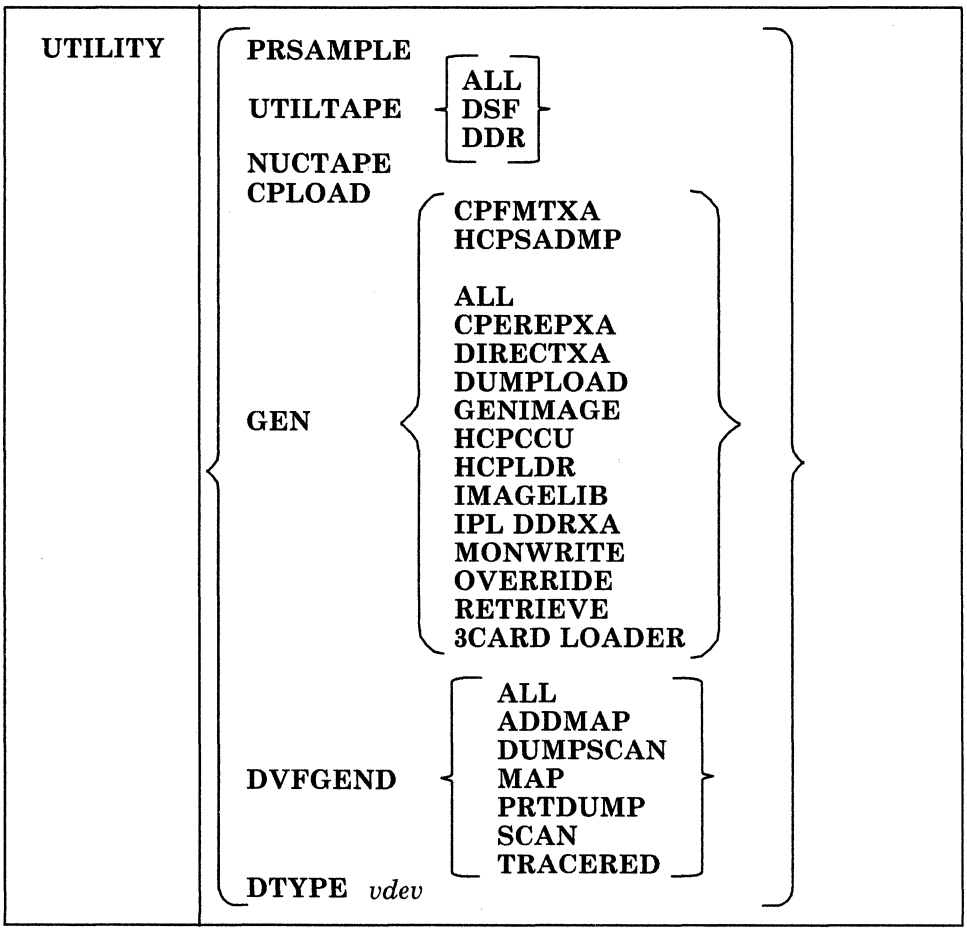
The UTILITY EXEC provides occasionally-used installation utility functions:

- Printing the system definition files
- Creating a stand-alone service utility tape for either or both DSF and DDR
- Writing a backup IPLable copy of the CP nucleus to tape
- Creating CP load list
- Generating the following utilities:

ADDMAP	CPEREPXA	CPFMTXA	DIRECTXA
DUMPLOAD	DUMPSCAN	GENIMAGE	HCPCCU
HCPLDR	HCPSADMP	IMAGELIB	IPL DDRXA
MAP	MONWRITE	OVERRIDE	PRTDUMP
RETRIEVE	SCAN	TRACERED	3CARD LOADER

Format

The format of the UTILITY EXEC is:



where:

PRSAMPLE
prints the following system definition files:

- USER DIRECT
- DMSNGP ASSEMBLE
- HCPRIO ASSEMBLE
- HCPSYS ASSEMBLE.

These files are spooled to the system printer. The minidisk(s) containing these files must be accessed before using the PRSAMPLE operand.

UTILTAPE
creates a tape containing any or all of the following stand-alone utility programs:

- Device Support Facility (DSF)
- DASD Dump/Restore program (DDRXA).

Each IPLable program is written as a separate tape file, with one or more tape marks separating the programs. Therefore, when you IPL a program off the tape, you may have to IPL more than once to bypass the tape marks.

NUCTAPE
writes a backup IPLable copy of the CP nucleus to tape.

CPLOAD
creates the CP load list, which has a file name of CPLOAD and a file type of EXEC.

GEN
creates any of the following stand-alone utility programs on disk from their associated object modules (text decks):

- CPEREPXA MODULE, the error recording module
- CPFMTXA MODULE, the module that CP-formats minidisks for CP use
- DIRECTXA MODULE, the user directory program
- DUMPLOAD MODULE
- GENIMAGE MODULE
- HCPCCU MODULE
- CMS version of the system loader (HCPLDR MODULE) and the stand-alone version of the system loader (HCPLDR LOADER)
- Stand-alone dump program (HCPSADMP)
- IMAGELIB MODULE
- 370-XA version of the DASD DUMP RESTORE program (IPL DDRXA)
- MONWRITE MODULE
- OVERRIDE MODULE
- RETRIEVE MODULE
- 3CARD LOADER.

UTILITY GEN ALL creates all the programs listed **except** CPFMTXA MODULE and HCPSADMP.

DVFGEND
creates any or all of the following modules:

- ADDMAP MODULE
- DUMPSCAN MODULE
- MAP MODULE
- PRTDUMP MODULE
- SCAN MODULE
- TRACERED MODULE.

DTYPE *vdev*
issues a DIAGNOSE code X'24' for the specified virtual device address, uses the returned information to look up the DASD type in the $DASD$ CONSTS file, and returns that information to the terminal (if UTILITY DTYPE is issued from the terminal) or to the program stack (if UTILITY DTYPE is called by another EXEC).

Only VM/XA-installation-supported DASD types are represented in $DASD$ CONSTS. The information returned is a single line consisting of three fields:

- DASD type and model
- The proper representation of the given DASD model to the CPFORMAT program.

If the virtual device address is invalid, non-existent, or addresses an unsupported device type, a non-zero return code is displayed.

The UTILITY DTYPE function is intended primarily for use by installation-related utilities.

Usage Notes

1. If you specify ALL for the GEN option, UTILITY generates all of the utilities listed under the GEN option, except CPFMTXA MODULE and HCPSADMP. If you specify ALL for the DVFGEND option, UTILITY generates all of the utilities listed under the DVFGEND option.
2. UTILITY creates HCPLDR MODULE and LOADER by assembling the associated files. It generates all other modules from the associated text deck. The text deck must be on an accessed disk before UTILITY is run. You must run SETUP EXEC, then VMFSETUP EXEC with the CP ALL parameters, to establish the appropriate minidisk access order.

Messages and Return Codes

DMSWUT002E **File** *fn ft* **not found**

Return Code: 28

DMSWUT050E **Parameter missing after** *value*

Return Code: 24

DMSWUT070E **Invalid parameter** *parameter*

Return Code: 24

DMSWUT095E **Invalid address** *vaddr*

Return Code: 24

DMSWUT338E **Generating a new** *fn ft* **will not be attempted**

Return Code: 28

DMSWUT340E *file name* **has not been created**

Return Code: 24

DMSWUT971E **Unable to locate** *fn ft*

Return Code: 24

DMSWUT972I *fn ft fm* **spooled the printer**

Return Code: 0

DMSWUT973R **Enter the minidisk address where the IPL decks were loaded**

Return Code: 0

DMSWUT974I **Unable to find IPL decks on the minidisk you indicated**

Return Code: 0

DMSWUT975I **Moving** *fn ft* **to tape**

Return Code: 0

DMSWUT976I **The** *fn ft* **program is on tape file number** *number*

Return Code: 0

DMSWUT980I **An IPLable CP nucleus now exists on tape**

Return Code: 0

DMSWUT986I **Unable to restore ACCESS to** *mdisk*

Return Code: 0

For a complete explanation of each message, refer to *VM/XA SP System Messages and Codes Reference*.

VMFAPPLY EXEC

Use the VMFAPPLY EXEC to create auxiliary control files for the PTFs you want to apply to VM/XA System Product.

Format

The format of the VMFAPPLY EXEC is:

VMFAPPLY	*prodid* [*compname*] [*updateid*] [(*options* ... [)]] *options:* [PUT / COR] [SETUP / NOSETUP] [EXCLUDE / NOEXCLUDE] [CHECK / NOCHECK] [LOG / NOLOG]

where:

prodid
is the product to be serviced. The *prodid* for VM/XA System Product Release 2 is 56643082.

compname
is the component to be serviced. The names for the VM/XA System Product Release 2 components are CP, CMS, GCS, and DV.

updateid
is the file type of the auxiliary control files. It should match a file type listed in the CNTRL file. If you do not specify an update ID, VMFAPPLY will use the update ID in the product parameter file.

options

PUT
indicates that you are applying program update service. PUT is the default.

COR
indicates that you are applying corrective service.

SETUP
specifies that a new minidisk access order will be set up. SETUP is the default.

NOSETUP
specifies that a new minidisk access order will not be set up. You must set the correct access order up yourself.

EXCLUDE
means that PTFs listed in the exclude list will not be applied. EXCLUDE is the default.

Note: If two PTFs are in the same text deck, you cannot exclude the first unless you also exclude the second, even if you list the first one on the exclude list. This restriction applies because the first PTF is considered to be a prereqisite for the second PTF.

If you list the first PTF in the exclude list, but not the second, and if the second PTF appears in the apply list, both PTFs will be applied. You will receive an error message indicating that a PTF you wanted to exclude has been applied.

NOEXCLUDE
means that the exclude list will be ignored.

CHECK
means that all the checks that can be specified in the product parameter file will be performed.

NOCHECK
means that none of the checks that can be specified in the product parameter file will be performed.

LOG
specifies that error messages will be written both to the apply exception log and, if critical, to the terminal. LOG is the default.

Notes:

1. Non-critical messages are written only to the apply exception log, not to the terminal.
2. Messages issued by the VMFOVER and VMSETUP EXECS, which VMFAPPLY invokes, are written only to the terminal, not to the apply exception log.

NOLOG
specifies that error messages will be written only to the terminal.

Note: Options specified on the VMFAPPLY command override the options specified in the product parameter file.

How VMFAPPLY Works

Figure 47 on page 674 is an overview of VMFAPPLY processing.

Verifying the Apply List

First, VMFAPPLY checks that the apply list is an appropriate apply list for program update service if you specified PUT, or for corrective service if you specified COR. If it is the wrong apply list, a message is issued.

Creating the Temporary PPF

VMFAPPLY calls VMFOVER to create a copy of the appropriate component section of the product parameter file with overrides. If the product has more than one component and no component was specified, you are shown a list of components and asked to choose one. This copy is used instead of the original product parameter file.

User Exit

A user exit EXEC call for setup and cleanup is provided. The user EXEC is identified in the PPF by the :USEREXIT. tag. If an EXEC name follows the :USEREXIT. tag, that EXEC is called at the beginning and end of the VMFAPPLY EXEC. A separate EXEC can be provided for each component. VMFAPPLY passes parameters indicating the receive function and either setup or cleanup, and a return code from VMFAPPLY. Examples of such calls might be:

```
MYEXEC VMFAPPLY SET-UP
MYEXEC VMFAPPLY CLEAN-UP rc
```

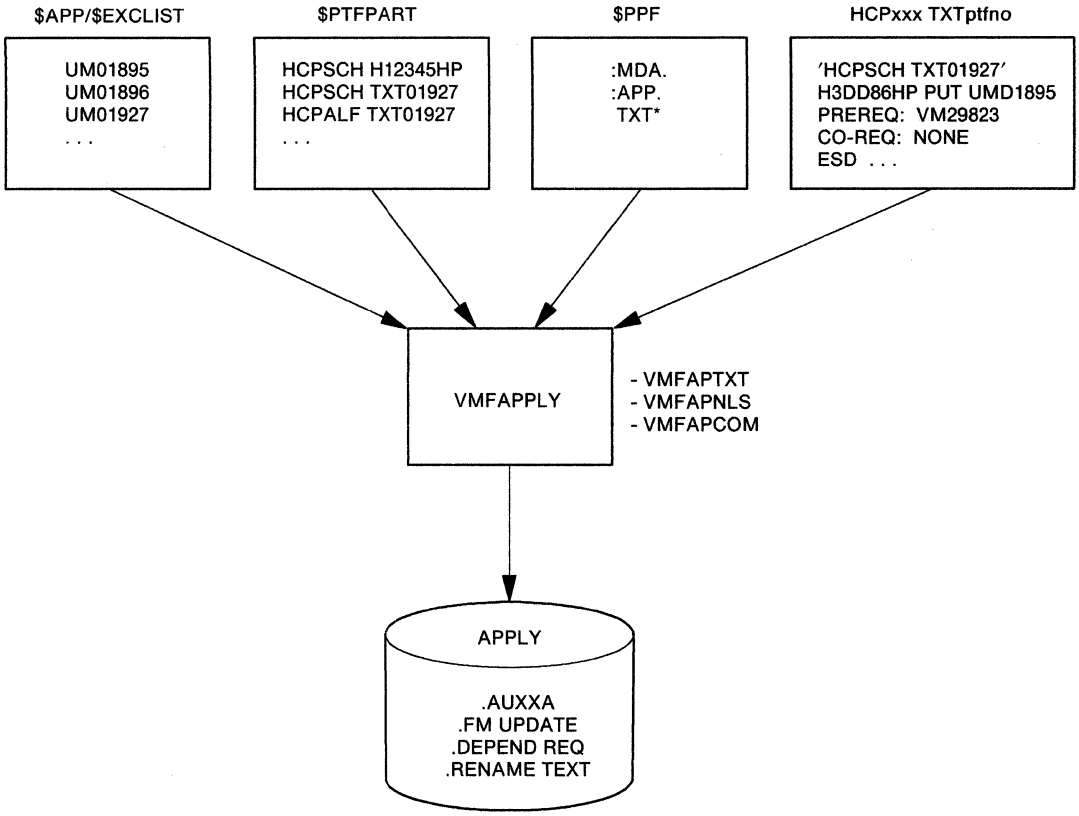


Figure 47. How VMFAPPLY Works

Minidisk Accesses

VMFAPPLY invokes VMFSETUP (unless you specified NOSETUP) to save your current disk access order and build a new disk access order based on the product parameter file. (The product parameter file should be on a system disk accessed in your current search order, preferably your A-disk.) The new search order is:

Task disk(s)
Apply disk(s)
Delta disk(s)
Local disk(s)
Base disk(s)
Build disk(s)
System disk(s).

The PTF Parts List

VMFAPPLY obtains the first PTF entry from the apply list. If the entry is not on the exclude list, VMFAPPLY locates the PTF parts list by using the PTF number (seven characters) for file name and $PTFPART for file type. VMFAPPLY works through the PTF parts list one at a time and calls the specific part handler based on entries in the PPF part type list.

After the PTF is applied, the file mode of the PTF parts list is changed to file mode 5. The procedure is repeated for each entry in the apply list.

If the :CKUPD tag in the product parameter file is set to NO, and the file mode of the PTF parts list is 5, no processing is performed on the PTF.

Part-Specific EXECs

VMFAPPLY invokes four part-specific EXECs (called part handlers). These part handlers are invoked once per part listed in the PTF parts list. The file type of the part determines which part handler is used.

1. VMFAPAUX supports AUX files in the apply function.
2. VMFAPTXT supports text parts in the apply function.
3. VMFAPNLS supports text parts that require national language support in the apply function.
4. VMFAPCOM is a common EXEC that insures the existence of parts that do not have specific processing but are listed in the PTF parts list.

VMFAPAUX EXEC: VMFAPAUX is invoked once for each AUX file in each $PTFPART list processed during the execution of VMFAPPLY.

VMFAPAUX locates the cumulative AUX file that contains the given PTF number. If this file does not exist, VMFAPAUX creates it. VMFPAUX validates the AUX file by commenting out duplicates which exist at lower AUX file levels. (These comments are located in the newly created file). If duplicate AUX files exist at higher levels, VMFAPAUX issues a message in the apply exception log.

VMFAPAUX adds included PTFs to the $APPINCL list, adds depend entries to requisite update files, adds requisite update files that don't have a file mode of 5 to the $APPREQ list, and changes the file mode of the update file to 5.

If the file contains new service, VMFAPAUX copies the new AUX file to a file with the file type listed in the :UPDTID tag of the PPF.

VMFAPCOM EXEC: VMFAPCOM is called to locate parts named in the PTF parts list for which no other specific processing is provided. Any part not found is listed in the exception log. The file mode is not changed to 5 because this is not a true apply (that is, no AUX structure is built) and there is no corresponding VMFBLD step.

VMFAPTXT EXEC: VMFAPTXT first looks for an existing AUX file (for the specified module and at the specified update level) in the APPLY disk sequence. If the AUX file exists on the current apply disk, it is used as a base and additional entries are added to it. If the existing AUX file is on an earlier disk in the sequence, it is copied onto the current disk and used as a base. If none exists, a new file is created.

VMFAPTXT gets the latest APAR entry (at the bottom of the text deck prolog) and checks the file mode of the update file. If the file mode number is 5, the APAR has already been applied. No further processing is required for that deck. If the file mode is 1, the entry is inserted at the top of the list of AUX file entries and the file mode number is changed to 5. If the update file is not found, it is created and a message is inserted in the apply exception log.

The update files should be found on either the DELTA or the APPLY disks. If they exist on the DELTA disk, source service is available. If they are on the APPLY disk, they are OCO modules with object service only. Changing the file mode from *x*1 to *x*5 indicates that the APAR has been applied (listed in an AUX structure). All applied OCO updates have the file mode of the APPLY disk; and all applied source updates have the file mode of the DELTA disk. All updates that have not been applied, but that do exist on the DELTA or APPLY disk, have the file mode *x*1.

The names (derived from APAR numbers) of update files in which prereq and coreq entries appear are entered in the requisite update files as DEPEND entries if the prerequisites and corequesites are in the same component as the update being applied. This will facilitate the backout of a bad fix and avoid leaving a dependent fix without its requisites. If the requisite file is not found, an entry is made in the apply exception

log. Requisites (prereq, coreq and/or ifreq) that are outside the component will be put in a list to be stored in the apply exception log.

VMFAPTXT proceeds to process all entries (working from bottom to top) in the text deck prolog until an already-applied entry (one whose file mode is 5) is encountered. Each succeeding entry from the text deck is inserted behind the last entry moved from the current text deck, but ahead of any entries that may have come from previous text decks. If the PTF number in any included entry is not in the apply list (minus the exclude list), a message is put in the apply exception log.

If CKTXT is specified in the PPF, and if the top AUX entry is being applied at this time, (that is, not previously applied as indicated by file mode 5 on the update file that was listed in the text deck prolog) VMFAPTXT checks to see if the text deck is a full text deck or a shell only. If it is a shell only, the module name is added to the apply exception log as a missing text deck. If it is a full text deck, VMFAPTXT checks to see if this full text deck supersedes any listed in the apply exception log as missing text decks, in which case the module name is purged from the log. VMFAPTXT reads and writes directly to the log on the user's A-disk.

After all entries have been processed, a search is made for AUX files at higher control file entries. AUX files at these CNTRL file entries may represent local modifications or local text patches. The AUX file file name and file type are listed in the apply exception log along with all entries from the AUX file.

Finally, VMFAPTXT searches the build lists in the BLD section of the PPF for the text deck file name. If the file name of the text deck being processed is found, a message is be put in the $VMFAPP $ERRLOG indicating what build list the text deck was found in. This allows the user to easily determine what parts can be built using VMFBLD.

After processing is complete, control returns to VMFAPPLY.

If the name is not found in the list, the deck is copied with the rename file type from the third column of the PPF entry in the apply section to allow compatibility with current MODULE build support. If a deck with the rename file type already exists, the self-documenting prologs are compared to determine the superset deck that should contain all entries from the other deck. The superset deck is kept with the renamed file type either by retaining the existing renamed deck or by copying the new deck with the REPLACE option.

After the copy, VMFAPTXT returns to VMFAPPLY.

VMFAPNLS EXEC: VMFAPNLS first creates an AUX file name by concatenating the text file name with a 1- or 2-character NLS suffix based on the text file type prefix. If the file type prefix is TAM, the language is mixed-case American English and the suffix is null. All other NLS file name suffixes are based on the file VMFNLS LANGLIST S, which is resident on MAINT's 190 disk and shown in Figure 48.

```
*   AMENG  TAM  American English
A   KANJI  TKA  Kanji
B   UCENG  TUC  Uppercase English
C   PORTG  TPO  Brazilian Portuguese
D   FRANC  TFR  French
E   GER    TGR  German
```

Figure 48. VMFNLS LANGLIST File

VMFAPNLS then looks for an existing AUX file (with the AUX file name created in the previous step and the specified update identifier) in the APPLY disk sequence. If the AUX file exists on the current APPLY disk, it is used as a base and additional entries are added to it. If the existing AUX file is on an earlier disk in the sequence, it is copied onto the current disk and used as a base. If none exists, a new file is created.

VMFAPNLS gets the latest APAR entry (at the bottom of the text deck prolog) and checks the file mode of the update file. If the file mode number is 5, the update has already been applied. No further processing is required for that deck. If the file mode is 1, the entry is inserted at the top of the list of AUX file entries and the file mode number is changed to 5. If the update file is not found, it is created and a message is inserted in the apply exception log.

The update files should be found on either the DELTA or the APPLY disks. If they exist on the DELTA disk, source service is available. If they are on the APPLY disk, they are OCO modules with object service only. Changing the file mode from *x*1 to *x*5 indicates that the APAR has been applied (listed in an AUX structure). All applied OCO modules have the file mode of the APPLY disk; and all applied source modules have the file mode of the DELTA disk. All modules that have not been applied, but that do exist on the DELTA or APPLY disk, have the file mode *x*1.

The names (derived from APAR numbers) of update files in which prereq and coreq entries appear are entered in the requisite update files as DEPEND entries if the prerequisites and corequesites are in the same component as the update being applied. This will facilitate the the backout of a bad fix and avoid leaving a dependent fix without its requisites. If the requisite file is not found, an entry is made in the apply exception log. Requisites (prereq, coreq and/or ifreq) that are outside the component will be put in a list to be stored in the apply exception log.

Messages indicating a requisite outside the component have a prefix of RO. These are not error messages, but informational messages issued so that you are sure to find all the requisites.

VMFAPNLS proceeds to process all entries (working from bottom to top) in the text deck prolog until an already-applied entry (one whose file mode is 5) is encountered. Each succeeding entry from the text deck is inserted behind the last entry moved from the current text deck, but ahead of any entries that may have come from previous text decks. If the PTF number in any included entry is not in the apply list (minus the exclude list), a message is put in the apply exception log, and the PTF is added to a supplemental apply list (called *fn* $APPINCL) to be processed after the main apply list.

If CKTXT is specified in the PPF, and if the top AUX entry is being applied at this time, (that is, not previously applied as indicated by file mode 5 on the update file that was listed in the text deck prolog) VMFAPNLS checks to see if the text deck is a full text deck or a shell only. If it is a shell only, the module name is added to the apply exception log as a missing text deck. If it is a full text deck, VMFAPNLS checks to see if this full text deck supersedes any listed in the apply exception log as missing text decks, in which case the module name is purged from the log. VMFAPNLS reads and writes directly to the log on the user's A-disk.

After all entries have been processed, a search is made for AUX files at higher control file entries. AUX files at these CNTRL file entries may represent local modifications or local text patches. The AUX file file name and file type are listed in the apply exception log along with all entries from the AUX file.

Finally, VMFAPNLS copies the deck with the NLS file type indicated in the PPF. If a deck with the NLS file type already exists, the self-documenting prologs are compared to determine the superset deck that should contain all entries from the other deck. The superset deck is kept with the NLS file type either by retaining the existing NLS text deck or by copying the new deck with the REPLACE option.

After the copy, VMFAPNLS returns to VMFAPPLY.

Return Codes

VMFAPPLY issues the following return codes:

Return Code	Explanation
0	No errors were encountered.
4	Minor errors were encountered. Check $VMFAPP $ERRLOG.
8	There was syntax error in the command.
24	Unexpected tag in PPF.
28	The required file was not found.
100	An unrecoverable error was encountered.

VMFBLD EXEC

Use the VMFBLD EXEC to build the nucleus during installation and to rebuild the nucleus after applying service. VMFBLD invokes $VMFPAT$, which invokes the HCPLDR command.

Format

The format of the VMFBLD EXEC is:

VMFBLD	*prodid* [*compname* / CORcompname] [*bldlist*] [(*options* ... [)]] *options:* [SETUP / NOSETUP] [CHECK / NOCHECK] [LOG / NOLOG] [DISK / PUNCH / TAPE]

where:

prodid
is the product to be serviced. The *prodid* for VM/XA System Product Release 2 is 56643082.

compname
is the component whose nucleus is to be rebuilt.

If you are applying corrective service, type COR*compname*.

The names for the VM/XA System Product Release 2 components are CP, CMS, GCS, and DV.

bldlist
is the loadlist or buildlist for the nucleus or module you wish to rebuild. If you do not specify *bldlist*, all entries for the component will be processed.

options

SETUP
specifies that a new minidisk access order will be set up.

NOSETUP
specifies that a new minidisk access order will not be set up. You must set the correct access order up yourself.

CHECK
means that all the checks that can be specified in the product parameter file will be performed.

NOCHECK
means that none of the checks that can be specified in the product parameter file will be performed.

LOG
specifies that error messages will be written both to the build exception log and, if critical, to the terminal.

Notes:

1. Non-critical messages are written only to the build exception log, not to the terminal.
2. Messages issued by the VMFOVER and VMSETUP EXECs, which VMFBLD invokes, are written only to the terminal, not to the build exception log.

NOLOG
specifies that error messages will be written only to the terminal.

DISK
specifies that the nucleus will be written to the target disk specified in the product parameter file. This must be the same disk specified on the SYSRES macro in the HCPSYS file (for CP) or the DMSNGP profile (for CMS).

PUNCH
specifies that the nucleus will be spooled to your virtual card reader.

TAPE
specifies that the nucleus will be written to tape 0182 in card image format.

Note: Options specified on the VMFBLD command override the options specified in the product parameter file.

How VMFBLD Works

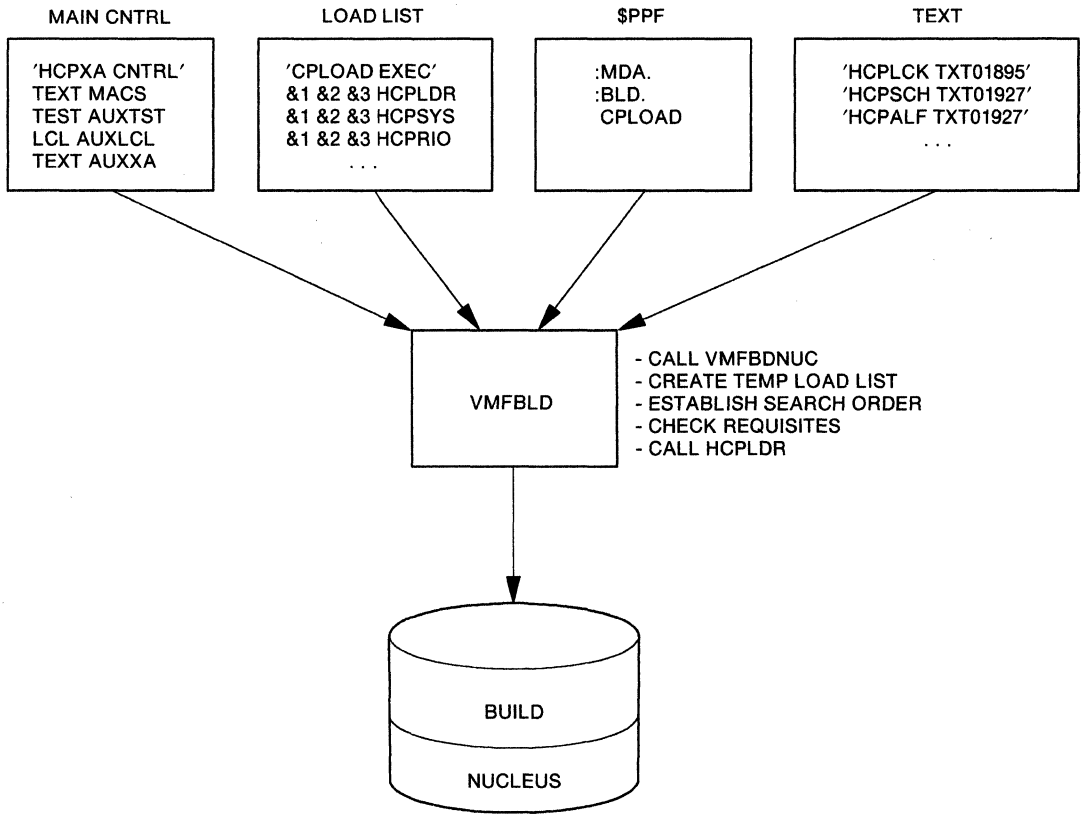


Figure 49 (Part 1 of 2). How VMFBLD Works

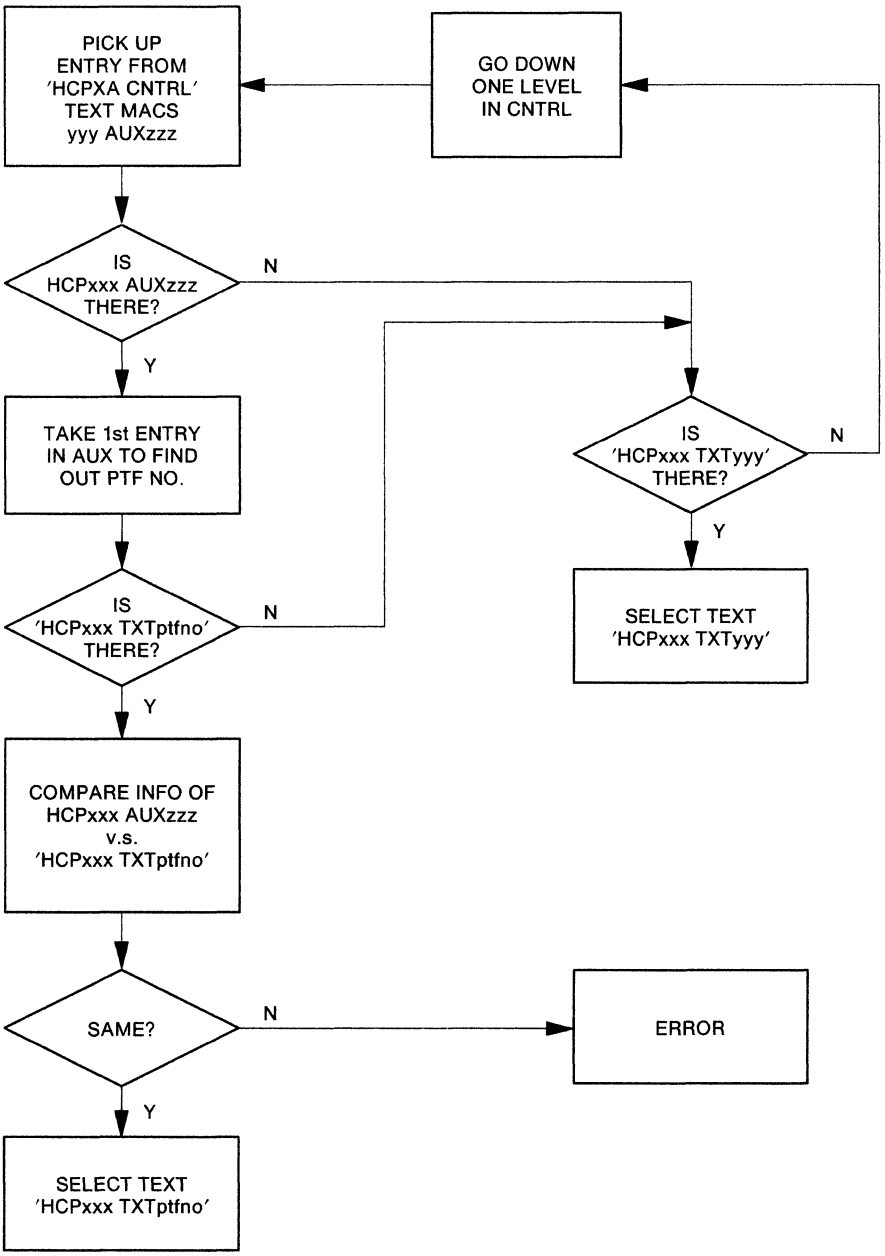


Figure 49 (Part 2 of 2). How VMFBLD Works

Creating the Temporary PPF

VMFBLD calls VMFOVER to create a copy of the appropriate component section of the product parameter file with overrides. This copy is used instead of the original product parameter file.

User Exit

A user exit EXEC call for setup and cleanup is provided. The user EXEC is identified in the PPF by the :USEREXIT. tag. If an EXEC name follows the :USEREXIT. tag, that EXEC is called at the beginning and end of the VMFBLD EXEC. A separate EXEC can be provided for each component. VMFBLD passes parameters indicating the receive function and either setup or cleanup, and a return code from VMFBLD. Examples of such calls might be:

```
MYEXEC VMFBLD SET-UP
MYEXEC VMFBLD CLEAN-UP rc
```

Minidisk Accesses

VMFBLD then invokes VMFSETUP (unless you specified NOSETUP) to save your current access list and establish a new list based on the product parameter file. The new access order is:

Task disk(s)
Local disk(s)
Apply disk(s)
Delta disk(s)
Base disk(s)
Build disk(s)
System disk(s).

Part-Specific EXECs

VMFBLD invokes the following part-specific EXECs:

- VMFBDNUC creates a temporary load list and invokes the system loader.
- VMFBDCPY copies the PTF-numbered TEXT deck to the BUILD disk.

VMFBDNUC EXEC: VMFBDNUC creates a new (temporary) load list named $$$TLL$$ EXEC that contains both file name and file type to pass to HCPLDR. VMFBDNUC will call the VMFLDS MODULE to perform a search for the TEXT decks to load and create the temporary load list.

When the temporary load list has been completed, the list of requisites within the component is processed as follows if the CHKREQ option has been specified. The list of requisite APARs is created by the VMFLDS MODULE.

1. VMFBDNUC creates actual update file types by combining the numeric portion of the APAR number with the system and version level from the PPF. For example:

APAR number	VM12345
System and version (and alternate)	S/DK A/XX
Numeric portion of APAR number	12345
Possible update file types	S12345DK A12345XX

2. VMFBDNUC creates a list of update files by file name and file type. For example:

   ```
   listfile * S12345DK * ■
   MODABC S12345DK
   MODXYZ S12345DK
   listfile * A12345XX * ■
   MODDEF A12345XX
   ```

3. VMFBDNUC locates MODABC, MODXYZ and MODDEF in the temporary load list and obtains the file type (PTF level) of the text deck included in the nucleus build.
4. VMFBDNUC searches the self-documenting prolog of the text deck to be sure that the APAR number (in this example, 12345) is included. If not, a message is put in the build exception log.

After requisite checking has completed, VMFBDNUC calls the $VMFPAT$ EXEC with the new load list containing file names and file types of text to be loaded. $VMFPAT$ looks for patches that need to be applied to text decks in the load list. Any patches found are inserted into the appropriate text deck and included in the IPL deck produced by the loader.

When the loader has completed punching the text decks, control returns to VMFBLD.

VMFBDCPY EXEC: VMFBDCPY will be used to copy the PTF-numbered TEXT deck to the BUILD disk identified in the PPF. VMFBDCPY calls VMFLDS to create a new (temporary) load list named $$$TLL$$ EXEC, which contains both the file names and file types of text decks to load. Requisite checking is performed as in VMFBDNUC. VMFBDCPY then invokes the COPYFILE command to copy the PTF-numbered deck to the BUILD disk and to rename it. These files must reside on the BUILD disk with a file type of TEXT.

Return Codes

VMFBLD issues the following return codes:

Return Code	Explanation
0	No errors were encountered.
4	Minor errors were encountered. Check VMFBLD $ERRLOG.
8	There was a syntax error in the command.
24	Unexpected tag in PPF.
28	The required file was not found.
100	An unrecoverable error was encountered.

VMFHASM EXEC

The VMFHASM EXEC updates a specified source file according to entries in a control file, and assembles the updated source file. VMFHASM invokes the CMS UPDATE command, then assembles the updated source file using Assembler H Version 2. If you wish, VMFHASM can use the product parameter file to establish the minidisk search order.

Format

The format of the VMFHASM EXEC is:

```
VMFHASM  fn [ctlfile                ]  [(options ... [)]]
            [prodid  [compname]     ]

         options:

         [SETUP  ] [DISK  ] [TERM   ] [LIST  ]
         [NOSETUP] [PRINT ] [NOTERM ] [NOLIST]

         [CTL ] [DECK    ] [RENT   ] [EXP] [XREF]
         [PPF ] [NODECK  ] [NORENT ]
```

where:

fn
is the file name of the source file to be updated. It must have a file type of ASSEMBLE.

ctlfile
is the file name of the control file. The control file must have a file type of CNTRL. If a product parameter file exists with a file name of *prodid*, the name of the CNTRL file will be found in the product parameter file under "Compname." You do not have to specify the ctlfile parameter.

prodid
is the product identifier. The *prodid* for VM/XA System Product Release 2 is 56643082.

compname
is the component name specified in the product parameter file. The names for the VM/XA System Product Release 2 components are CP, CMS, GCS, and DV.

options

SETUP
specifies that a new minidisk access order will be set up.

NOSETUP
specifies that a new minidisk access order will not be set up. You must set the correct access order up yourself.

DISK
places the listing file on a virtual disk.

PRINT
writes the listing file to the printer.

TERM
writes the diagnostic information on your display. The diagnostic information consists of the diagnosed statement followed by the error message issued.

NOTERM
suppresses the TERM option.

LIST
produces an assembler listing.

NOLIST
does not produce an assembler listing.

CTL
indicates that the second parameter is a control file name.

PPF
indicates that the second parameter is a product parameter file name. If neither CTL nor PPF is specified, the VMFHASM EXEC searches for a file whose name is the same as the second parameter. If such a file is found, its file type determines whether it is a control file or a product parameter file. If both exist, the product parameter file is used.

DECK
creates an object module.

NODECK
suppresses the DECK option.

RENT
checks the program for a possible violation of program reenterability. An error message identifies code that makes the program nonreenterable.

NORENT
suppresses the RENT option.

EXP
prints all macro expansions. By default, macro expansions are not printed.

XREF
causes the XREF(FULL) option to be invoked when VMFHASM invokes the assembler. The default for VMFHASM is XREF(SHORT).

How VMFHASM Works

The steps taken by the VMFHASM EXEC are summarized below:

1. If a product parameter file is used, VMFHASM calls VMFOVER to create a copy of the appropriate component section of the product parameter file with overrides. This copy is used instead of the original product parameter file.

   VMFHASM then invokes VMFSETUP (unless you specified NOSETUP) to save your current disk access order and build a new disk access order based on the product parameter file.

2. The VMFHASM EXEC calls the UPDATE command, which includes the CTL, STK, and PRINT options.

   UPDATE uses the control file (*ctlfile* CNTRL) to update the assembler language source file. The new file is named $*fn* ASSEMBLE.

   UPDATE stacks information from the control file in the console stack, and prints the update log file.

3. Using the library list from the MACS record in the control file, VMFHASM issues a GLOBAL MACLIB command.

4. The updated source file, $*fn* ASSEMBLE, is assembled using the options indicated on the VMFHASM command line.

5. The output text deck from the assembly, $*fn* TEXT, is concatenated with the UPDATES file so that the text deck contains a history of update activity.
6. UPDATE stacks the identifier of the most recent update it found and applied. (If no updates were found, UPDATE stacks the identifier of the MACS record in the main control file instead.) If no PTF number or local tracking number is available, VMFHASM uses this update level identifier to determine how to rename $*fn* TEXT.

   If a product parameter file was specified (or used by default), and if the update was listed in an auxiliary control file containing the PTF number or local tracking number, the text deck is renamed *fn xxxnnnnn*, where *xxx* is the text file type prefix from the corresponding record in the main control file and *nnnnn* is the PTF number or local tracking number. The default text file type prefix is TXT.

   If a control file was specified (or used by default); or if the PTF number or local tracking number is not available, and if the update level identifier is TEXT, the text deck is renamed *fn* TEXT. If the update level identifier is anything other than TEXT, the text deck is renamed *fn* TXT*xxxxx* (where *xxxxx* is the 1- to 5-character update level identifier).
7. Temporary files ($*fn* ASSEMBLE and *fn* UPDATES) are erased.

The input and output files used by VMFHASM are summarized below:

Input and Output Files

Disk Input Files

fn ASSEMBLE	Assembler language source
fn CNTRL	Control file
fn MACLIB	Macro library
fn AUX*xxxxx*	Auxiliary control files
fn UPDT*xxxx*	Update files
fn $PPF	Product parameter file

Disk Output Files

fn {TEXT|TXT*xxxxx*}	Object deck, named according to the update level identifier in the control file

This file also contains data from the UPDATES file, together with date and time information. It is placed on the A-disk, which you must access in read/write mode before issuing VMFHASM.

Printer Output Files

$*fn* LISTING	Assembler listing (if PRINT option is in effect)

This file also contains data from the update log file (fn UPDLOG), describing the updates applied to the source file.

Messages

The UPDATE command issues messages to indicate when each update file is applied.

ASMBLING *fn*

Explanation: The assembly is going to begin. If you specified any assembler option on the VMFHASM command line, the options used are also displayed.

fn {**TEXT**|**TXT***xxxxx*} **CREATED**

Explanation: This message indicates the file name and file type of the text deck.

Note: If the source is on some other R/W minidisk instead of the A minidisk, VMFHASM won't create a text deck identifying the update level on the A minidisk.

VMFMAC EXEC

The VMFMAC EXEC updates macro libraries. It invokes the CMS UPDATE command to update specified copy[1] or macro files, according to entries in a control file, and then builds a new macro library from the resulting new versions of those files.

Format

VMFMAC	*libname ctlfile*

where:

libname
is the file name of the macro library to be updated, and of the EXEC file that contains the names of the library members. The entries in *libname* EXEC must be in the following format:

&1 &2 *fn1*
&1 &2 *fn2*

⋮

where *fn1*, *fn2*, and so on, are file names of macro or copy files to be updated and included in the macro library, which must have a file type of MACLIB.

ctlfile
is the file name of a control file to be used to apply the updates. The file type must be CNTRL.

How VMFMAC Works

1. VMFMAC locates *libname* EXEC and the control file. It also erases any existing files named NEWMAC MACLIB and NEWMAC COPY. Then VMFMAC begins reading the macro or copy file names from the EXEC, beginning at the bottom.
2. For each entry in the *libname* EXEC, VMFMAC:
   - Invokes the UPDATE command with its CTL option to apply the updates specified in the control file
   - Adds the updated macro or copy file (\$*file name* MACRO or \$*filename* COPY) to the macro library NEWMAC MACLIB
   - Adds the UPDATES file created by the UPDATE command to the file NEWMAC COPY
   - Erases \$*filename* MACRO or \$*file name* COPY, and file name UPDATES.
3. If there are no update files corresponding to a macro or copy file entry specified in *libname* EXEC, the macro or copy file is added to NEWMAC MACLIB in its current form. NEWMAC COPY, containing a history of updates applied by VMFMAC, is added to NEWMAC MACLIB.

---

[1] Copy files are collections of assembler statements that many modules use in common. In source listings, the assembler includes copy files by a "COPY *copyfilename*" statement. They are similar to macro definitions because they reside in libraries, but they contain no substitution data. VM/XA System Product uses copy files to define control block DSECTS, data areas, and tables.

4. If no errors occur during the procedure, then when all the macros have been added to NEWMAC MACLIB, NEWMAC MACLIB is renamed *libname* MACLIB. Then, *libname* MACLIB, if it exists, is erased.

   If errors occur during the VMFMAC EXEC procedure (for example, if a MACRO or a COPY file is not found) *libname* MACLIB is not erased, and the updated macro library retains the name NEWMAC MACLIB.

Input and Output Files

Disk Input Files

libname EXEC	List of macro or copy files to be updated and/or included in *libname* MACLIB.
ctlfile CNTRL	Control file used by the UPDATE command.
fn MACRO *fn* COPY ⋮	Files to be updated and/or included in the macro library.
auxiliary control files	
update files	

Disk Output Files

libname MACLIB	Updated macro library.
libname COPY	Contains the UPDATE files produced by UPDATE command processing.

Printer Output Files

The printer is spooled with the CONT option, so that when VMFMAC completes, the printer file contains:

- A copy of the control files
- The update log file for each updated macro or copy file the UPDATE command produced
- A copy of each macro or copy file in the macro library
- The *libname* COPY file, which contains the accumulated UPDATES files the UPDATE command created.

Notes:

1. When VMFMAC adds files with MACRO file types to a MACLIB, the EXEC takes the membername from the macro prototype statement. When VMFMAC adds files with COPY file types to a MACLIB, the EXEC follows these rules:

   a. If the membername is from the COPY file and updates were found, the membername is $*filename*.
   b. If the membername is from the COPY file and no updates were found, the membername is *filename*.
   c. If you include a *COPY statement as the first record in the file, the membername is that designated on the *COPY statement. The format of the *COPY statement is:

      ```
      *COPY membername
      ```

2. If errors occur during VMFMAC processing, consult the NEWMAC COPY file printed by VMFMAC. If you can correct the errors involving one or two macro or copy files, you can use the MACLIB

command to add these members to NEWMAC MACLIB. Then erase the current *libname* MACLIB and rename NEWMAC MACLIB *libname* MACLIB.

Messages

The UPDATE command issues the message DMSUPD178I to inform you of the updates being applied to each macro or copy file. If no updates are found, message DMSUPD181E is issued.

fn {**COPY|MACRO**} **ADDED**

Explanation: The specified macro or copy file has been added to the macro library.

libname **COPY ADDED**

Explanation: *libname* COPY, containing the update history of the MACLIB, has been added to the macro library.

VMFLKED EXEC

The VMFLKED EXEC procedure invokes the CMS LKED command to link-edit modules into a LOADLIB. VMFLKED uses the normal CMS search order when searching for TEXT files.

Format

The format of the VMFLKED EXEC is:

VMFLKED	*fn* [*ft* / **LKEDCTRL** [*fm* / *]] [(*options*. . .[ )]]

where:

fn
is the file name of the input control file. You must specify a file name.

ft
is the file type of the input control file. The default file type is LKEDCTRL.

fm
is the file mode of the input control file. The default file mode is *.

PRINT
prints out a hardcopy of the linkage-editor output.

MODULE *module_name*
indicates that only those members of the LOADLIB that include *module_name* are to be link-edited.

How VMFLKED Works

VMFLKED reads the specified LKEDCTRL file and expects to find option records (if any are needed) followed by linkage-editor input records. Multiple groups of options followed by linkage-editor input can be combined in a single LKEDCTRL file (see Figure 50 on page 694).

1. VMFLKED processes the option records, which are identified by a percent sign (%) in column one.
2. VMFLKED processes linkage-editor input records when it finds a nonoption and noncommentary record.

   When VMFLKED finds:

   - An INCLUDE record, then the process adds the record to the linkage-editor input file and issues a FILEDEF for the TEXT file
   - A NAME record, then the process adds the record to the linkage-editor input file and invokes the linkage editor
   - A commentary record (beginning with *), then the process ignores the record
   - Any other record, then the process adds the record to the linkage-editor input file.
3. VMFLKED continues with Step 1 to process the next group of option records (if any).

Input and Output Files

Input Files

fn TEXT — The name of the TEXT file. When the process finds an INCLUDE statement in the linkage-editor input control file, it issues a FILEDEF for that TEXT file. The linkage editor reads this TEXT file.

fn LKEDCTRL — A modified linkage-editor file. INCLUDE cards include TEXT files, and you cannot use them to include files from a library. Any record containing an asterisk (*) in column one is commentary and the process ignores it.

The VMFLKED EXEC also recognizes special control (option) records in the linkage-editor control file (see Table 19). These records, which must begin with a percent sign (%) in column one, may only be located between linkage-editor input files (that is, groups of them may be at the beginning of the file or following a NAME record).

Table 19 (Page 1 of 2). Linkage-Editor Control File Special Control (Option) Records

Record	Parameter	Function
%CONTROL	*control_file_name*	This record indicates to take the file types for linkage-editor input files from a control file. The name of the control file is specified on the %CONTROL record and the file type is CNTRL. When the %CONTROL record is read, VMFLKED reads the control file and uses the first "word" of each line of the file to build an array of file types. The first line of the file indicates the default file type. Each of the file types is checked and if it is not TEXT, then the characters TXT are put at the beginning. When an VMFLKED finds an INCLUDE card, it searches the array. VMFLKED uses the first file type from the array that matches an existing file. This option is in effect until a %NOCONTROL record is found.
%NOCONTROL		This record indicates not to take the file type for linkage-editor input files from a control file.
%LIBRARY	*load_library_name*	This record changes the LOADLIB to be used and the name on the LKEDIT listing. The default for the LOADLIB name is the file name of the LKEDCTRL file.
%LEPARMS	*link-edit parameters*	This record sets the link-edit parameters used in each link-edit step. If no parameters are specified, then none are used.
%MAXRC	*maximum_valid_return_code*	This record sets the maximum valid return code. The VMFLKED EXEC checks this value when it ends. If the highest return code is higher than this value and is not listed on the %ACCEPTRC record, then the EXEC issues a warning message.

Table 19 (Page 2 of 2). Linkage-Editor Control File Special Control (Option) Records

Record	Parameter	Function
%ACCEPTRC	*return code(s)*	This record lists the acceptable return code(s) for VMFLKED processing. (The return code(s) are usually higher than entry on the %MAXRC entry). VMFLKED checks the value(s) after each link-edit. If the return code(s) after any link-edit is higher than the %MAXRC record and not specified on the $ACCEPTRC record, then VMFLKED issues a warning message at the end of its processing.
%IGNORE		This record causes the EXEC to bypass the warning message for missing TEXT files. Once specified, this record takes effect for all subsequent link-edits (or until a %NOIGNORE record is found).
%NOIGNORE		This record causes a warning message to be issued if there is a missing TEXT file. Once specified, this record takes effect for all subsequent link-edits (or until a %IGNORE record is found).
%ERASE		This record erases LOADLIB and LKEDIT files. This is useful when you rebuild the LOADLIB as it keeps the LOADLIB and LKEDIT files as small as possible. If this record is not specified, then the LOADLIB and LKEDIT files are not erased.

Note: The %ERASE control statement takes effect immediately, and erases whatever is the current LOADLIB. The LOADLIB is not erased if you use the MODULE option. (The LOADLIB can be changed using the %LIBRARY control statement).

Output Files

fn LOADLIB — The main output from the linkage editor. This file contains the link-edited load modules.

fn LKEDIT — The file that contains the linkage-editor map for all modules.

The following is an example of a LKEDCTRL file.

```
%CONTROL YOURCTRL
%LIBRARY NCCF
%ERASE
%MAXRC 4
%ACCEPTRC 12 14
%LEPARMS NCAL LIST XREF LET RENT
  INCLUDE  DSIZDST
  INCLUDE  DSIZSHP
  INCLUDE  DSILUTRM
  ORDER    DSIZDST
  ENTRY    DSIZDST
  NAME     DSIZDST(R)
%LEPARMS NCAL LIST XREF LET REUS
  INCLUDE  DSIVMCMD
  INCLUDE  DSIVMMSG
  ORDER    DSIVMCMD
  ENTRY    DSIVMCMD
  NAME     DSIVMCMD(R)
%IGNORE
%LEPARMS NCAL LIST XREF LET RENT
  INCLUDE  DSIPRTVM
  ENTRY    DSIPRTVM
  NAME     DSIPRTVM(R)
```

Figure 50. Example of a LKEDCTRL file

Messages

DMSWLK002E **File** *fn ft* [*fm*] **not found**
DMSWLK002W **File** *fn ft* **not found**
DMSWLK003E **Invalid option** *opt*
DMSWLK005E **No {LKEDCTRL|MODULE} specified**
DMSWLK010E **Premature EOF on file** *fn ft*
DMSWLK065E **{PRINT|MODULE} option specified twice**
DMSWLK842E **No {control|library} file name found in** *fn ft* [*fm*]
DMSWLK843I **An invalid control record was found and ignored:**
DMSWLK844E **No linkedit performed**
DMSWLK845W **Errors were encountered during the link edit processing that will probably make the loadlib unusable**
DMSWLK846I **LKED** *target_module* **into** *library*

For a complete explanation of each message, refer to *VM/XA SP System Messages and Codes Reference.*

VMFMERGE EXEC

The VMFMERGE EXEC procedure applies PTFs (program temporary fixes) from the DELTA disk to the MERGE disk.

Do not use this procedure to service any of the base components of VM/XA SP. Use this procedure when applying PTFs to System Network Architecture (SNA) products.

VMFMERGE requires a service control file (*ptfnum* SCF) for each requested PTF and its requisites. The service control file contains instructions for applying the PTF. To use VMFMERGE, you must access the minidisk containing the `prodid` VMFPARM file. This file identifies the minidisks that VMFMERGE must access to service each product.

Format

The format of the VMFMERGE command is:

VMFMERGE	*prodid* **PTF** {*ptfnum* | * } [**EXCLUDE** *exclist*] **PTFLIST** *applist*

where:

prodid
is the ID of the product you specify.

You cannot specify a prodid of EXCLUDE, because EXCLUDE is a keyword for this EXEC.

PTF {*ptfnum* / * }

applies a single PTF if you specify a PTF file name (*ptfnum*). If you enter an asterisk (*) instead of a PTF file name, you will apply all PTFs for the product, as listed in the apply list, called *prodid* APPLIST, supplied on the service tape. You cannot specify a file name of EXCLUDE, because EXCLUDE is a keyword for this EXEC.

If the service tape contains an exclude list file named *prodid* EXCLIST, any PTF listed in that file is not applied.

PTFLIST *applist*
applies the selected PTFs listed in the apply list file, *applist* APPLIST. If you specify a file name of EXCLUDE, you cannot use the EXCLUDE option to specify an Exclude List.

If the service tape contains an exclude list file named *prodid* EXCLIST, any PTF listed in that file is not applied.

EXCLUDE *exclist*
excludes the selected PTFs listed in the exclude list file, *exclist* EXCLIST. PTFs listed in `prodid` EXCLIST are excluded, in addition to those in the user-specified exclude list. If you specify EXCLUDE as the apply list file name, you cannot use the EXCLUDE option to specify an exclude list.

If you use *prodid* EXCLIST as the name of your exclude list, then VMFMERGE ignores any other *prodid* EXCLIST file, including the one that may be supplied on the service tape, during processing. You should therefore use some other name for the exclude list you create.

Note: If two PTFs are in the same text deck, you cannot exclude the first unless you also exclude the second, even if you list the first one on the exclude list. This restriction applies because the first PTF is considered to be a prereqisite for the second PTF.

If you list the first PTF in the exclude list, but not the second, and if the second PTF appears in the apply list, both PTFs will be applied. You will receive an error message indicating that a PTF you wanted to exclude has been applied.

How VMFMERGE Works

VMFMERGE is a service process that processes PTFs. The EXEC procedure:

1. Uses the parameter file (`prodid` VMFPARM) to determine the virtual address of the merge and delta disks.
2. Checks the merge log to insure that the PTF you select is not already merged or superseded.
3. Reads the service control file to get the prerequisite and corequisite PTFs and the elements affected by this PTF.
   - If the service control file for any of the prerequisite or corequisite PTFs is missing, then processing for the current PTF stops.
   - If a prerequisite or corequisite PTF is in the exclude list, then the current PTF is not merged.
   - If a prerequisite or corequisite PTF has a status of "SUPERSEDED", then that prerequisite or corequisite is not merged.
   - If the requisite is not within this product, the system displays a message indicating that the requisite PTF must be merged at some later time.
   - When you merge a PTF that is a requisite of a change in another product, be sure to note this requisite information. There is no automatic way to tell you of this cross-product requisite if at a later time you remove the change that is a requisite of a change in another product.
4. Does the necessary COPY/RENAME from the delta disk to the merge disk for each element in the prerequisite and corequisite chain that was not "SUPERSEDED" or already merged.

   Note: Temporary files are created during this COPY/RENAME process to insure system integrity.

   These files are erased during normal VMFMERGE processing.
5. Adds service history to the element if it is a text deck. Element history consists of text deck comment(s) containing:
   - The PTF or APAR number
   - A time and date stamp
   - Any apartext information that was in the service control file (SCF).

   Note: A temporary file ($APARTXT $VMFMERG) is created during this history process to ensure system integrity. VMFMERGE erases this file during normal processing.
6. Updates the reqby log to reflect all the requisite relationships of all the merged and superseded PTFs.
7. Marks in the merge log any ZAPs and PTFs that are superseded by this PTF. Those ZAPs and PTFs are never applied.
8. Puts entries in the merge log to show which PTFs have been merged.

Notes:

1. You can issue VMFMERGE to process a single PTF, a list of PTFs, or all PTFs on the input disk for a product.
2. Merged PTFs cannot be excluded, but can be superseded.
3. Superseded PTFs cannot be merged nor excluded.
4. PTFs in the Exclude List can be superseded.

Messages

DMSWMG002E **File** *fn ft [fm]* **not found.**
DMSWMG008E **Device** *vdevno* **invalid or nonexistent.**
DMSWMG017E **Invalid device address** *vdevno.*
DMSWMG520E **Invalid operand:** *operand.*
DMSWMG545E **Missing operands.**
DMSWMG649E **Extraneous parameter** *parm.*
DMSWMG653E **Error executing** *command.*
DMSWMG856E **Disk address** *vdevno* **is listed more than once on the DELTA and/or MERGE entry records in the** *prodid* **VMFPARM file.**
DMSWMG857E **The number of disk addresses on the DELTA entry record cannot exceed nine.**
DMSWMG858E **Unable to find a** *tag* **entry record in the** *fn ft* **file.**
DMSWMG859E **The** *prodid* **VMFPARM file has no disk addresses on the {MERGE|DELTA} entry record.**
DMSWMG860E **Only one {MERGE|DELTA} entry record may appear in the** *prodid* **VMFPARM file.**
DMSWMG861I **Accessing** *disk_type* **disk** *vdevno* **as** *fm.*
DMSWMG862I **Change** *name* **has been** *action.*
DMSWMG863E **The MERGE disk** *vdevno* **must be linked read-write.**
DMSWMG864E **PTF** *name* **will not be** *action* **because it already is** *status.*
DMSWMG864I **PTF** *name* **will not be** *action* **because it already is** *status.*
DMSWMG864W **PTF** *name* **will not be** *action* **because it already is** *status.*
DMSWMG865I **Processing PTF** *name.*
DMSWMG866W **No PTFs have been** *action.*
DMSWMG867E **Invalid status** *status* **in** *prodid* **VMFMGLOG for entry** *ptf.*
DMSWMG868E **PTF** *name* **is not a part of product** *prodid.*
DMSWMG869E **Error in file** *fn ft fm. data* **is invalid for** *tag* **tag.**
DMSWMG870E **Error in file** *fn ft fm.* **There are no elements.**
DMSWMG871E **Error in file** *fn ft fm.* **The** *name* **tag is missing.**
DMSWMG872E **Error in file** *fn ft fm.* **REPLACE tag missing after the element** *name.*
DMSWMG873E **Error in file** *fn ft fm. parm* **is an invalid parameter. Expecting parameter(s) PRODID, PREREQ, COREQ, SUP, APARTEXT, or CHANGES.**
DMSWMG874E **Invalid entry found at line** *line* **in** *fn ft.*
DMSWMG882W **File** *fn ft [fm]* **[from** *name*] **not found on any DELTA disks from the VMFPARM file.**
DMSWMG883W **PTF** *name* **is not a part of product** *prodid* **and must be** *action* **in product** *prodid.*
DMSWMG886E **File name** *name* **from the** *fn ft [fm]* **file is longer than 8 characters.**
DMSWMG892E **PTF** *name* **has not been** *action.*
DMSWMG893E **Incomplete processing, not all [required] PTFs were** *action.*
DMSWMG893W **Incomplete processing, not all [required] PTFs were** *action.*
DMSWMG898E **VMFREMOV processing is incomplete.**

For a complete explanation of each message, refer to *VM/XA SP System Messages and Codes Reference.*

VMFNLS EXEC

The VMFNLS EXEC procedure updates national language-related files. VMFNLS automatically applies updates to source files, generates text files, and renames them so they can be loaded into the system. If you have your own language files, this EXEC is provided so that you can maintain those files.

Format

VMFNLS	*fn ft* {*ppfname* [*compname*] / *ctlfile*} [(*options* [])]] *options:* [PPF / CTL] [SETUP / NOSETUP]

where:

fn
is the file name of the source file that is to be converted to text.

ft
is the file type of the source file that is to be converted to text. Only ASSEMBLE, REPOS, and DLCS file types are allowed.

ppfname
is the file name of either the base product parameter file (which is the product identification number, *prodid*) or a product parameter override file containing an override area for the component. The file type must be $PPF.

compname
is the name of either a base component area (which is a base component name, such as CP or CMS) or a component override area containing alternative and/or additional parameters for the component. If you do not specify a valid name, VMFNLS shows you a list of names and asks you to enter one.

ctlfile
is the file name of the control file that is used to apply updates to the source file before text is generated. The file type must be CNTRL. The IBM-supplied control files are described in Chapter 7, "How VM/XA System Product Uses Control Files and Update Files" on page 377.

options

PPF
indicates that the third operand in the command is the name of a product parameter file or override file that specifies the minidisk/directory search order and the name of the control file.

CTL
indicates that the third operand in the command is the name of a control file. If CTL is specified, the product parameter file is not used.

If neither CTL nor PPF is specified, VMFNLS searches for a file whose name matches the third operand in the command. If a file with that name is located, VMFNLS checks the file type to determine if the file is product parameter file ($PPF) or a control file (CNTRL). If both files exist, the product parameter file is used.

SETUP
sets up a minidisk/directory access order for the assemble function. This option is valid only when using a product parameter file.

NOSETUP
does not set up a new access order.

Note: In addition to the options described above, VMFNLS accepts certain options for three of the commands that it issues:

- ASSEMBLE
- GENMSG
- CONVERT COMMANDS.

Some options are assigned by VMFNLS; other options that can be used depend on the file type of the source file. See the following section, "How VMFNLS Works." For complete descriptions of these commands and their options, see *VM/XA SP CMS Command Reference.*

How VMFNLS Works

1. If a product parameter file is being used:
   a. VMFNLS calls the VMFOVER EXEC to build a temporary product parameter file (file type $PPFTEMP) containing the parameters for the component being processed.
   b. If a user EXEC is specified in the USEREXIT tag in the $PPFTEMP file, VMFNLS calls the user EXEC with the parameters VMFNLS and SET-UP.
   c. If the NOSETUP option is not specified, VMFNLS calls the VMFSETUP EXEC with the ACCESS ASM options to save the current minidisk/directory access order and build a new one for the assemble function using data from the $PPFTEMP file.
2. VMFNLS issues the UPDATE command with the CTL, STK, PRINT, and OUTMODE A1 options.

   UPDATE uses the control file (*ctlfile* CNTRL) specified in the product parameter file or in the VMFNLS command to update the assembler language source file.

   UPDATE stacks information from the control file in the console stack and prints the update log file.

   The OUTMODE A1 option specifies that the files created by the UPDATE command are written on file mode A with a file mode number of 1 (A1).

   If a PTF file is missing, UPDATE writes a message to the console and to the update log file.
3. At this point, processing depends on the file type of the input source file:
   - If the source file is an ASSEMBLE file:
     a. Using the library list from the MACS record(s) in the control file, VMFNLS issues a GLOBAL MACLIB command.
     b. VMFNLS issues the ASSEMBLE command to assemble the updated source file.
   - If the source file is a REPOS (message repository) file:
     a. VMFNLS determines the *langid* associated with the source file:
        - If the source file name is only six characters, VMFNLS assigns AMENG (American English) as the default *langid.*
        - If the source file name is longer than six characters, VMFNLS extracts the country code from the seventh and eighth characters of the file name and searches the VMFNLS LANGLIST file to find the corresponding *langid.* This file contains a list of all valid country codes, along with the associated *langids* and language names, as shown in Figure 51 on page 700.

```
*   AMENG  TAM  American English
A   KANJI  TKA  Kanji
B   UCENG  TUC  Uppercase English
C   PORTG  TPO  Brazilian Portuguese
D   FRANC  TFR  French
E   GER    TGR  German
```

Figure 51. VMFNLS LANGLIST File

If the country code from the source file name is not contained in the VMFNLS LANGLIST file, then all temporary files are erased, the printer is closed, and VMFNLS terminates.

b. VMFNLS issues the GENMSG command to compile the input REPOS file and produce a text file. VMFNLS sets up the operands for GENMSG as follows:

fn
is the file name of the file to be compiled. GENMSG places this name in the ESD card when the text file is produced. The possible file names are:

fn	**Repository**
HCPMES	CP
DMSMES	CMS
CSIMES	GCS

ft
is REPOS, the file type of the repository file to be compiled.

applid
corresponds to the first three characters of the file name of the source file:

applid	**Application**
HCP	CP
DMS	CMS
CSI	GCS

langid
is the language identifier of the source file.

options
depend on the component associated with the source file. For the CP repository, VMFNLS uses the GENMSG options `CP MARGIN 63`. For other components, VMFNLS uses the GENMSG option `MARGIN 63`.

Note: For a Kanji CP repository, VMFNLS uses the GENMSG options `CP DBCS`. For other Kanji repositories, VMFNLS uses the GENMSG option `DBCS`.

Other GENMSG options can be specified with the VMFNLS command if they do not conflict with the GENMSG options assigned by VMFNLS.

GENMSG produces a text file that has the same file name as the input file and a file type of TEXT. GENMSG also produces a LISTING file, whose file name is the source file's file name with a prefixed dollar sign.

The following table shows some examples of the text and listing files written to file mode A from a given message repository source file:

Source File	Text File	Listing File
HCPMES REPOS	HCPMES {TEXT|TXT*nnnnn*}	$HCPMES LISTING
DMSMESA REPOS	DMSMESA {TEXT|TXT*nnnnn*}	$DMSMESA LISTING
CSIMESB REPOS	CSIMESB {TEXT|TXT*nnnnn*}	$CSIMESB LISTING

Note: *nnnnn* is the number of the most recently applied PTF.

The updated source file is then printed.

For a complete description of the GENMSG command, see *VM/XA SP CMS Command Reference*.

- If the source file is a DLCS (definition language for command syntax) file, VMFNLS issues the CONVERT COMMANDS command to produce two text files from the input file.

  VMFNLS sets up the operands for CONVERT COMMANDS as follows:

  fn
  is the file name of the file to be compiled.

  ft
  is the file type of the file to be compiled.

  fm
  is always set to *.

  options
  are not assigned by VMFNLS; any valid CONVERT COMMANDS options can be specified.

  The file names of the text decks produced by CONVERT COMMANDS depend on the DLCS statement contained in the input file. This statement identifies the *applid*, *langid*, and whether the input file is a user or system DLCS file. CONVERT COMMANDS uses the *langid* to obtain the corresponding country code (*y*) from the VMFNLS LANGLIST file.

  For a system DLCS file, the file names of the text decks are *applid* SPA*y* for the command syntax definition file and *applid* SSY*y* for the translation and synonym table. For a user DLCS file, the file names of the text decks are *applid* UPA*y* for the command syntax definition file and *applid* USY*y* for the translation and synonym table.

  The file type of the output text files is TEXT.

  The following table shows some examples of the text files produced from a given source file:

Source File	Text Files
DMSSPA DLCS	DMSSPA TEXT
	DMSSSY TEXT
DMSSPAB DLCS	DMSSPAB TEXT
	DMSSSYB TEXT
DMSSPAC DLSC	DMSSPAC TEXT
	DMSSSYC TEXT

  The updated source file is then printed.

  For a complete description of the CONVERT COMMANDS command, see the *VM/XA SP CMS Command Reference*.

4. Each output TEXT file from the ASSEMBLE, GENMSG, or CONVERT COMMANDS processing is concatenated with the UPDATES file so that the TEXT file contains a history of update activity.
5. VMFNLS renames the TEXT file as follows:
   - If the PPF option is specified and no PTF prefix is found in the control file, the TEXT file is renamed *fn* TXT*nnnnn*, where *nnnnn* is the PTF number obtained from last AUX file entry processed by the UPDATE command. If a PTF prefix is found in the control file, that prefix overrides the TXT default.
   - If the CTL option is used, VMFNLS uses the update level identifier from the control file (the identifier of the most recent update that was found and applied is stacked by the UPDATE command), to rename the TEXT file. If the update level identifier is anything other than TEXT, the TEXT file is renamed *fn* TXT*nnnnn* (where *nnnnn* is the 1- to 5-character update level identifier).

   The new *fn* TXT*nnnnn* or *fn* TEXT file is found on file mode A.

   Note: The new text deck will be file mode A1 regardless of the file mode of the original text deck. If the file mode number of the original text deck is anything other than 1, you must rename the text deck created by VMFNLS.
6. If a product parameter file is being used:
   a. VMFNLS calls VMFSETUP to restore the original minidisk/directory access order.
   b. If a user EXEC is specified in the USEREXIT tag in the $PPFTEMP file, VMFNLS calls the user EXEC with the parameters VMFNLS, CLEAN-UP, and a return code.

Messages

DMSWNL001E **No file name specified.**
DMSWNL002E **VMFNLS LANGLIST * not found.**
DMSWNL023E **No file type specified.**
DMSWNL032E **Invalid file type** *ft*.
DMSWNL122E **Return code** *rc* **from** *routine*.
DMSWNL179E **Missing or invalid MACS card in control file** *fn ft fm*.
DMSWNL328E **Control file not specified.**
DMSWNL448E **Country code** *code* **not in** *list*.

For a complete explanation of each message, see *VM/XA SP System Messages and Codes Reference*.

VMFOVER EXEC

The VMFOVER EXEC makes a temporary copy of one component section from the product parameter file with changes made in accordance with the override section of the product parameter file or a separate PPF override file. The other service and installation EXECs use this copy instead of the base product parameter file.

VMFREC, VMFAPPLY, VMFHASM, and VMFBLD all issue VMFOVER automatically. You can also issue VMFOVER from the command line if you want to see the temporary product parameter file.

Format

The format of the VMFOVER EXEC is:

VMFOVER	*prodid compname*

where:

prodid
is the product identifier of the product to be serviced. For VM/XA System Product Release 2, the product identifier is 56643082.

compname
is the component to be serviced, for example, CP or CMS. *compname* can be a name listed after the :COMPLST. tag or the :OVERLST. tag in the base PPF; or after the :OVERLST. tag in the PPF override file. If you do not specify the *compname*, you will be shown a list of components and asked to choose one.

How VMFOVER Works

VMFOVER locates the file with a file name of *prodid* and a file type of $PPF. This file can be either a base PPF or a PPF override file. The EXEC then checks the list of component names following the :COMPLST. tag for the component name you specified. If the name is found, this is the base product parameter file. The appropriate component section is copied to a file called *prodid* $PPFTEMP, and processing stops.

If the component name is not listed after the :COMPLST. tag, VMFOVER looks for it after the :OVERLST. tag. If the component name is not listed there, VMFOVER issues an error message.

If the component name is found after the :OVERLST. tag, VMFOVER looks for a *:compname.* tag. The *:compname.* tag tells VMFOVER where to look for the basic information about the component. For example:

```
:CP. CP 56643082

:CORCP. CP
```

The first example is from an override file. It tells VMFOVER that the basic information about CP is found in the CP section of 56643082 $PPF. The second example is from the override section of a base PPF. Notice that it does not specify a file name. It tells VMFOVER to look for the base information about CORCP in the CP section of *this file*.

Override information follows the *:compname.* tag. VMFOVER combines the override information with the base information to create a copy of the appropriate component section of the product parameter file. The file name of the copy is the file name of the override file if the NEWNAME option was specified on the *:compname.* tag in the override file; otherwise, it is the file name of the base PPF. The file type of the copy is $PPFTEMP.

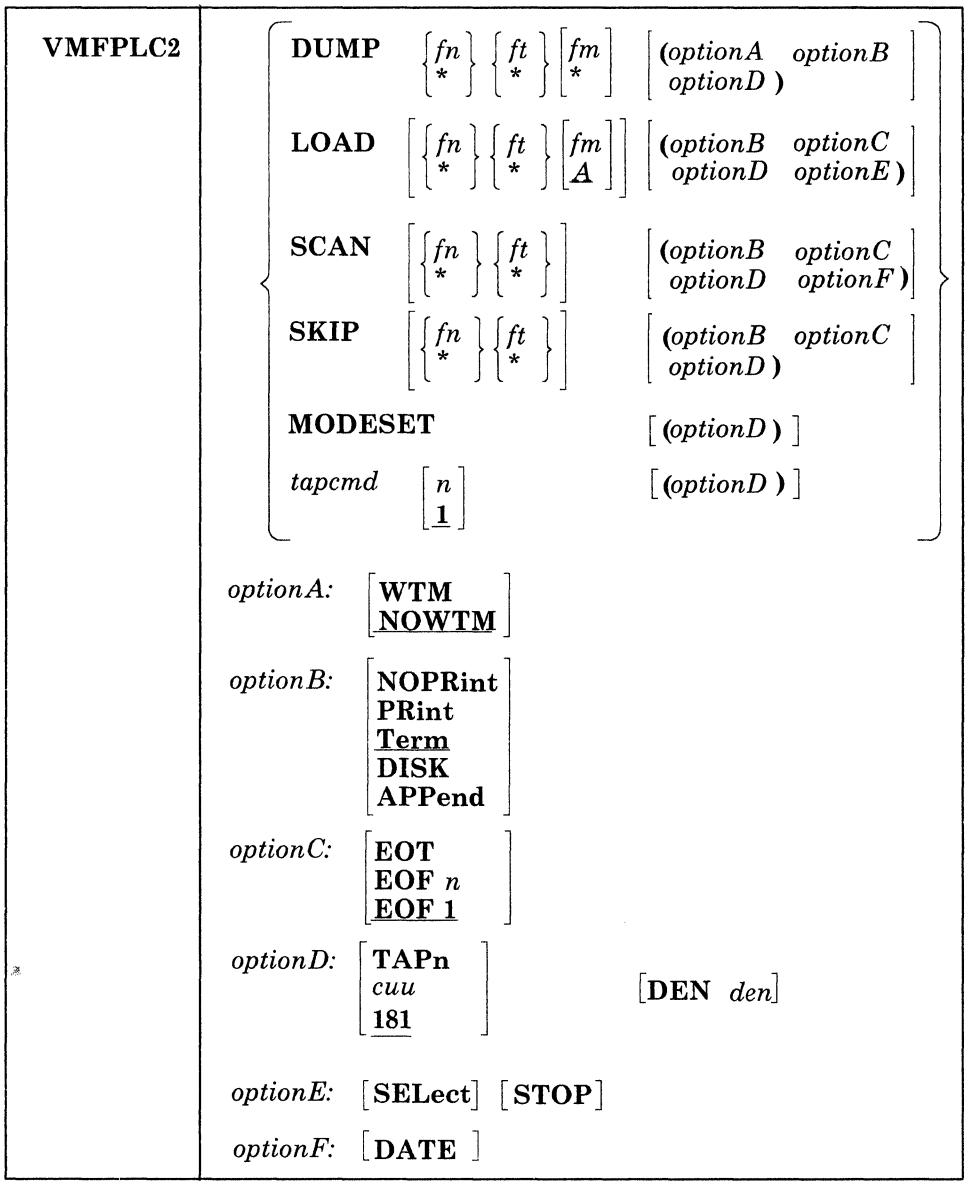
For examples of a base product parameter file with an override section, an override file, and $PPFTEMP files, see Chapter 8, "Files Used in Program Update Service and Corrective Service."

VMFPLC2 Command

The VMFPLC2 command loads source code files from the Product Tape and loads the service installation EXEC (VMSERV) from the program update tape. VMFPLC2 also dumps CMS files from disk to tape, loads previously dumped files from tape to disk, and performs various control operations on a specified tape drive. Disk files to be dumped can contain either fixed-length or variable-length records.

The VMFPLC2 command does not process multivolume files. Files processed by the VMFPLC2 command must be in a unique CMS format.

Format



where:

DUMP $\left\{\begin{matrix}fn\\*\end{matrix}\right\}\left\{\begin{matrix}ft\\*\end{matrix}\right\}\left\{\begin{matrix}fm\\*\end{matrix}\right\}$

dumps one or more disk files to tape. If *fn* or *ft* is specified as an asterisk (*), all files that satisfy the other file identifier are dumped.

If *fm* is coded as a letter, that disk and its extensions are searched for the specified file(s). If *fm* is coded as a letter and number, only files with that mode number and letter (and the extensions of the disk referenced by that *fm* letter) are dumped. If *fm* is coded as an asterisk (*), all accessed disks are searched for the specified file(s). If *fm* is not specified, only the A-disk and its extensions are searched.

LOAD $\left\{\begin{matrix}fn\\*\end{matrix}\right\}\left\{\begin{matrix}ft\\*\end{matrix}\right\}\left\{\begin{matrix}fm\\\underline{\mathbf{A}}\end{matrix}\right\}$

reads CMS files onto disk. The *optionC* options control the area on the tape that the command searches. For instance, if EOF 1 (the default) is in effect, VMFPLC2 loads only the file(s) specified before the first physical tape mark.

If a file identifier is specified, only that one file is loaded. If an asterisk (*) is specified for *fn* or *ft*, VMFPLC2 loads all files that satisfy the other file identifier.

The files are written to the disk indicated by the file mode letter. The file mode number, if entered, indicates that only files that have that file mode number are to be loaded.

SCAN $\left\{\begin{matrix}fn\\*\end{matrix}\right\}\left\{\begin{matrix}ft\\*\end{matrix}\right\}$

lists the identifiers of the files it scans. The *optionC* options control the area on the tape that the command searches. For instance, if EOF 1 (the default) is in effect, VMFPLC2 scans only the file(s) specified before the first physical tape mark. However, if a file identifier (*fn* and *ft*) is specified, scanning stops upon encountering that file; the tape remains positioned ahead of the file.

SKIP $\left\{\begin{matrix}fn\\*\end{matrix}\right\}\left\{\begin{matrix}ft\\*\end{matrix}\right\}$

lists the identifiers of the files it skips. The *optionC* options control the area on the tape that the command searches. For instance, if EOF 1 (the default) is in effect, VMFPLC2 skips only the file(s) specified before the first physical tape mark. However, if a file identifier (*fn* and *ft*) is specified, skipping stops after encountering that file; the tape remains positioned immediately following the file.

MODESET

sets the values specified by the DEN, TRACK, and TRTCH options. After initial specification in a VMFPLC2 command, these values remain in effect for the virtual tape device until they are changed in a subsequent VMFPLC2 command, RDTAPE, WRTAPE, or TAPECTL macro.

tapcmd $\left[\begin{matrix}n\\\underline{1}\end{matrix}\right]$

specifies a tape control function (*tapcmd*) to be executed *n* times (default is 1 if *n* is not specified). These functions also work on tapes in a non-CMS format.

Tapcmd	Action
BSF	Backspace *n* tape marks
BSR	Backspace *n* tape records
ERG	Erase gap
FSF	Forward-space *n* tape marks
FSR	Forward-space *n* tape records
REW	Rewind tape to load point
RUN	Rewind tape and unload
WTM	Write *n* tape marks

WTM
writes a tape mark on the tape after each file is dumped.

NOWTM
writes a tape mark after each file is dumped, then backspaces over the tape mark so that subsequent files written on the tape are not separated by tape marks.

NOPRINT
does not spool to the printer the list of files dumped, loaded, scanned, or skipped.

PRINT
spools to the printer the list of files dumped, loaded, scanned, or skipped.

TERM
displays a list of files dumped, loaded, scanned, or skipped.

DISK
creates a disk file containing the list of files dumped, loaded, scanned, or skipped. The disk file has the file identification of TAPE MAP A5.

APPEND
causes the disk file (containing the list of files dumped, loaded, scanned, or skipped), to be added to the end of an existing TAPE MAP.

EOT *n*
EOF **EOF 1**

EOT reads the tape until an end-of-tape indication. EOF *n* reads the tape through a maximum of *n* tape marks. EOF 1 is the default.

n
TAP*cuu*
181

specifies the virtual device address of the tape to be read or written to; *cuu* is 181, 182, 183, or 184. The default is 181. You can also use TAP*n*, where *n* is a single digit number. The number is appended to the number 18 to form a device number. For instance, TAP1 becomes 181 and TAP2 becomes 182. The unit specified by *cuu* must previously have been attached to your CMS virtual machine before any tape I/O operation can be attempted.

DEN *den*
is the tape density, where *den* is 800, 1600, 6250, or 38K. 9TRACK is assumed. In the case of either 800/1600 or 1600/6250 dual-density drives, 1600 is the default.

SELECT
inhibits loading of a file from the tape in cases when VMFPLC2 would replace an *identical* file on the disk. Files will be loaded only if they do not exist on the specified disk, or when the date/time stamp for the file on the disk does *not match* the date/time stamp for the corresponding file on the tape.

STOP
assumes that files contained on the tape are in alphabetical sequence. If the requested file is on the tape, the file is loaded onto disk and the tape stops. If the file is not on the tape and a file is encountered that is alphabetically beyond the bounds of the requested file, the tape stops. You must specify *fn ft*. Neither *fn* nor *ft* may be specified as an asterisk (*).

DATE
displays listfile information during a SCAN. The information displayed includes number of records, length of records, and date/time stamp.

Usage Notes

1. If conflicting options are specified on the VMFPLC2 command line, the last option entered on the command line is in effect.
2. The VMFPLC2 command writes tape records 4005 bytes long. The first character is a binary 2 (X'02'), followed by the characters CMS and a file format byte, followed by 4000 bytes of file data packed without regard for logical record length. If a null block is dumped, the character 0 replaces the byte after CMS. This causes subsequent loading of null blocks to be ignored. In the final record, the character N replaces the blank after CMS, and the data area contains CMS file directory information.
3. If a tape file contains more CMS files than would fit on a disk, the tape load operation may terminate if there is not enough disk space to hold the files. To prevent this, when you dump the files, separate logical files by tape marks, then forward space to the appropriate file.
4. The CMS file directory is the first record of each CMS file on tape.
5. It is possible to run a tape off the reel in at least one situation. If you specify EOF *n* and *n* is greater than the number of tape marks on the tape, the tape will run off the reel.
6. For more information on tape file handling, see *VM/XA SP Real System Operation.*
7. Do not use TAPE as a synonym of VMFPLC2, or vice versa. If you do, do not use these synonyms to call either function from within another EXEC. You may use any other valid synonym, for example, TAP.

VMFREC EXEC

Use the VMFREC EXEC to map the program update tape and to receive update files from the PUT tape.

Format

The format of the VMFREC EXEC is:

VMFREC	{INFO [(*options1* [*options2*]] }
	{ *prodid* [*compname* / *product exec parameters*] / LIST *fn ft* } [(*options2* ... [)]]
	options1: **MEMOS**|EXECS|SDMAP|ALL
	options2: [PUT / COR] [LOG / NOLOG] [SETUP / NOSETUP]

where:

INFO
creates the service disk map.

For all options of INFO, the service disk map and other files specified by the option are loaded to file mode C if it is accessed, or on file mode A if file mode C does not exist.

The following options are used with INFO:

options1:

MEMOS
loads the tape descriptor file, the tape document, and product memos. The service EXECs and PPF files are not loaded with this option. MEMOS is the default option.

EXECS
loads the service EXECs and PPF files from the first and second files on the tape.

SDMAP
creates the service disk map and loads the tape descriptor file.

ALL
loads the tape descriptor file, the tape documents, product memos, product parameter files and service execs.

prodid
is the product identifier of the product to be serviced. For VM/XA System Product Release 2, the product identifier is 56643082.

compname
is the component listed in the product parameter file to be serviced. The names for the VM/XA System Product Release 2 components are CP, CMS, GCS, and DV.

If you do not specify the *compname* for a product that has more than one component, you will be shown a list of components and asked to choose one.

product exec parameters
are any parameters you want to pass to the product-specific EXEC that VMFREC invokes.

LIST *fn ft*
gives the file name and file type of a file containing a list of product identifiers and, optionally, product EXEC parameters for products you wish to service.

If any product identified in the list supports the service enhancements, the product parameter file for that product must be loaded from the tape for the LIST option to be successful.

options2:

PUT
indicates that you are receiving files from the program update service tape. PUT is the default.

COR
indicates that you are receiving files from the corrective service tape.

LOG
specifies that error messages will be written both to the receive exception log and, if critical, to the terminal. LOG is the default.

Notes:

1. Non-critical messages are written only to the receive exception log, not to the terminal.
2. Messages issued by the VMFOVER EXEC, which VMFREC invokes, are written only to the terminal, not to the receive exception log.

NOLOG
specifies that error messages will be written only to the terminal.

SETUP
specifies that a new minidisk access order will be set up. SETUP is the default.

NOSETUP
specifies that a new minidisk access order will not be set up.

Note: The SETUP and NOSETUP options are not applicable to VMFREC INFO and its associated options.

Usage Notes

1. The default for VMFREC INFO is MEMOS, which ignores the EXECs in the first file on the tape and the PPFs in the second file on the tape.
2. You can replace the current level of the service EXECs on your system by using the EXECs option on the VMFREC INFO command. The EXECs option loads the service EXECs and product parameter files to file mode C if it exists, or to file mode A.
3. You can set up a separate minidisk for the service EXECs and product parameter files. To do this:
   a. Allocate and format the minidisk.
   b. Specify this minidisk in the TASK statement of your product parameter file as a read-only disk. For example:

   If you always have a disk accessed as the C-disk while receiving service, the A-disk will never have service-executable parts on it. Disk address 5E5 is used in this example.

```
work disk       a-disk 191
file 1 & 2      c-disk 5E5 service parts from tape
service disk    b-disk 5E5 read-only extension of itself during
                apply and build
```

The above access order can be used to access the 5E5 disk as file mode C immediately after file mode A during VMFREC INFO. This access order works in conjunction with the following override file:

```
:OVERLST. CP
:CP.   CP 56643082
:MDA.
TASK   5E5/           * replace existing system card with service
**                    * build disk as an extension of itself (read-only)
```

This allows the A-disk to be used for temporary replacement of service EXECs on the service build disk (the C-disk).

c. Change your SETUP EXEC to access only this minidisk.

To load the service EXECs and product parameter files to your service minidisk, access your service minidisk as file mode C and enter:

```
VMFREC INFO (EXECS
```

Note: The EXECS option of VMFREC INFO loads any product parameter files from tape file 2 of the tape volume currently mounted. If the service tape is a multi-volume tape, you must run `VMFREC INFO (EXECs` for all volumes that contain product parameters file of interest to you.

How VMFREC Works

The following lists summarize VMFREC processing: Figure 52 on page 717 illustrates VMFREC processing.

For VMFREC INFO:

1. VMFREC creates the tape map on file mode A and copies it to $LEVEL $TAPE file on file mode C (if it exists), or on file mode A. Then VMFREC maps the 3-character file type abbreviations to their real file types, creates the list of service parts found on the tape, and does service level checking.
2. VMFREC creates or adds to the service diskmap for this volume.

The rest of the steps decribe the actions taken by VMFREC for the MEMO, EXECs, SDMAP, and ALL options.

- For the MEMO option, VMFREC:
  1. Loads the documentation CMS files from tape files 1 and 2 to file mode C (if it exists) or file mode A.
- For the EXECs option, VMFREC:
  1. Saves the old service parts and notifies the user for each part saved.
  2. Loads the service-related CMS files from tape files 1 and 2.
  3. Copies the service files with PTF-numbered file types to files with the same names and real file types.
  4. Creates or updates $LEVEL EXEC.
- For the SDMAP option:
  1. Since the purpose of this option is to create the service disk map, the steps for this option are already done at the beginning of VMFREC processing.
- For the ALL option, VMFREC:
  1. Saves the old service parts and notifies the user for each part saved.
  2. Loads all the CMS files from tape files 1 and 2.
  3. Copies the service files with PTF-numbered file types to files with the same names and real file types.

4. Creates or updates $LEVEL EXEC.

For all other VMFREC options:

- When invoked with a *ppfname* or a *prodid*, either directly or through the LIST operand, VMFREC:
  - Creates the tape map on file mode A and copies it to the $LEVEL $TAPE file on file mode C (if it exists), or on file mode A
  - Maps the 3-character file type abbreviations to their real file types
  - Creates the list of service parts found on the tape
  - Checks service levels
  - Creates or adds to the service diskmap for this volume.
- Then VMFREC performs the following processing for the specified *ppfname* or *prodid* (or, if the LIST operand is used, for each product or component in the list):
  1. VMFREC determines whether the product or component uses a product parameter file. To do this, it looks at the file type of the level identifier (located in the second tape file).
     - If the file type is *crllnn1*, the product or component uses a product parameter file. The rest of this section describes the processing for products and components that use a product parameter file. The product parameter file from the highest-level minidisk in the search order is used to control the receive function.
     - If the file type is other than *crllnn1*, the product or component uses a product-specific EXEC. VMFREC does steps 5 and 6 page 712, then calls the product-specific EXEC and passes any parameters entered with the command or in the LIST file. For more information about product-specific EXECs, see the product documentation.
  2. VMFREC calls the VMFOVER EXEC to build a temporary product parameter file (file type $PPFTEMP) containing the parameters for the component being processed.
  3. If a user exit is specified in the USEREXIT tag in the $PPFTEMP file, VMFREC calls the user EXEC with the parameters VMFREC and SET-UP.
  4. If the SETUP option is taken, VMFREC accesses the minidisks and/or directories required to set up the following access order:

     TASK
     APPLY
     DELTA
     BASE
     BUILD
     SYSTEM.
  5. VMFREC checks the PUT/COR option specified with the command (PUT is the default) against the tape descriptor (located in the first tape file) to make sure that the correct tape is mounted for the type of service you want to do.
  6. VMFREC positions the tape to the service files for the specified product or component.
  7. VMFREC does merge processing as follows:
     a. VMFREC locates the MERGE tag in the $PPFTEMP file. The MERGE tag may list one or more names of minidisk strings, such as DELTA1 or LOCAL1.
     b. VMFREC then locates the record for the first string name in the MDA section of the $PPFTEMP file. This record lists the minidisks in the string in search order form from highest to lowest, left to right.

c. Beginning with the lowest search order pair in the string (the last two minidisks listed on the right), VMFREC:

   1) Calls the COPYFILE command with the REPLACE option and the OLDDATE option to copy all the files from the higher of the pairs in the search order to the lower.

   2) Erases the higher of the pair.

   3) Selects a new pair consisting of the higher of the previous pair and the next higher in the string.

   4) Repeats the above steps for the new pair.

   5) Continues selecting pairs until the highest minidisk in the string has been merged. This results in an empty minidisk at the highest level.

d. VMFREC look at the next string name (if any) in the MERGE record and repeats the merge processing for that string.

8. VMFREC calls one or more of the following part-specific EXECs to process the service tape and load the service files. The next tape file on the service tape contains a product contents directory that lists the service files. The RECSER section of the $PPFTEMP file identifies the part-specific EXEC that handles each type of file and specifies the target string. The MDA section of the $PPFTEMP file identifies the specific target minidisk in the string.

   VMFRCAXL This EXEC receives the apply lists and exclude lists.

   VMFRCAUX VMFRCAUX receives AUX files and renames the AUX file file types with the ISD abbreviation for the AUX file and the PTF number from the top entry of the AUX file.

   VMFRCUPD This EXEC receives source update files as follows:

   a. VMFRCUPD checks to make sure that the target does not already contain any update files for the component. If any update file exists, VMFRCUPD stops processing.

   b. VMFRCUPD searches the other accessed minidisks for files with the same file name and file type as the update files that have just been received.

      - If it finds an update shell with the same name, VMFRCUPD erases the update shell, but retains the dependent information and the file mode number.

      - If it finds a complete update file with the same name, VMFRCUPD erases the new file.

   VMFRCUPP This EXEC process uppercase English HELP files.

   VMFRCTXT This EXEC receives text decks as follows:

   a. VMFRCTXT checks to see if a text deck with the same file name and file type exists in the target string.

      - If a text deck with the same file name and file type exists on a previous level minidisk, the new deck is not received, and processing for that text deck ends. This prevents cumulative text decks received from a previous service tape from being duplicated in the data base.

      - If a text deck with the same file name and file type exists on the target minidisk, the new deck is received to replace the existing text deck. The existing text deck is probably is text shell created by VMFREC, because only one copy of a specific text deck is shipped on the service tape. If the existing text deck is a full text deck (due to receiving the same service tape

twice or receiving more than one service tape on a single target), no harm is done by the overlay.

b. VMFRCTXT checks the LTO (last text only) record in the $PPFTEMP file and processes the text deck as follows:

   - If LTO is set to NO, VMFRCTXT loads the text deck to the target.
   - If LTO is set to YES, VMFRCTXT compares the text deck prolog with the prologs of all other text decks that have the same file name and file type prefix (for example, TXT). VMFRCTXT keeps the text deck with the latest entries and creates text deck shells for the others. If the text deck you receive on the tape does not include all the entries from earlier text decks, VMFRCTXT writes a message to the receive exception log.

c. VMFRCTXT processes the history (prolog) section of all the text decks on the target minidisk or directory.

   - If a source update file does not exist for the text deck (because the part is serviced by replacement), VMFRCTXT creates an update shell on the APPLY minidisk.
   - If a text deck does not exist in the target string, VMFRCTXT creates a text deck shell.

VMFRCCOM This EXEC calls the VMFPLC2 command to receive update files without additional processing.

9. After any needed part-specific EXECs finish processing, control returns to VMFREC with a return code.
10. If a user EXEC is specified in the USEREXIT tag in the $PPFTEMP file, VMFREC calls the user EXEC with the parameters VMFREC and CLEAN-UP, and with a return code.

Level Checking

VMFREC EXEC does level checking to determine whether the level of service EXECs on the tape is the same as, or at a higher level than, the service EXECs on your system.

The $LEVEL MAP file

VMFREC INFO uses a file called $LEVEL MAP for service EXEC level checking. $LEVEL MAP is in the first tape file on the tape. The first line in the $LEVEL MAP is a comment that contains the tape type (PUT or COR) and the tape level. Two examples of this comment lines are:

```
* SES PUT LEVEL 8901

* SES COR LEVEL 8901A
```

In the second example, the A in 8901A indicates the level of service EXECs that is on the tape. In this example, the A indicates one change was made for COR tapes after PUT 8901.

Each entry after the comment contains four tokens. The first is the part file name, the second is the part executable file type, and the third and fourth are the part PTF numbered file types for each version of the service EXECs supported. The VM/XA SP Release 2 version is listed first. These last two columns allow a VM/SP Release 6 customer to map the VM/XA SP 2 PTF numbers to the corresponding VM/SP Release 6 PTF numbers. Unserviced parts are shipped with a file type of the three-character abbreviation followed by a PTF number of 00000 (for example, EXC00000).

An example of these entries follows.

```
* SES parts in the first tape file
*                XA/SP2  370/SP6
VMFREC   EXEC   EXC12345 EXC12346
VMFRCTXT EXEC   EXC12345 EXC12346
$VMFMSG$ EXEC   EXC23123 EXC23124
VMFLDS   MODULE MOD12345 MOD12346
```

Next in the $LEVEL MAP file are the PPF entries for the second file on the tape. Since there is a one-to-one relationship between PPF and product, the file name and file type are followed by only one column containing the PTF-numbered file type.

An example of the PPF entries follows.

```
* PPFs in the second tape file

56643082 $PPF   $PF12345
5664167E $PPF   $PF12346
```

The Level Checking Algorithm

VMFREC checks to ensure that a known level of the service tool is being used. It checks the following:

- Installed service tool level

  VMFREC always reports the level of the service tool installed on your system by displaying the level found in the first record of the $LEVEL EXEC file. If no $LEVEL EXEC file exists then the level is reported as UNKNOWN.

- Tape level, against installed level

  The service tool parts in files 1 and 2 of the service tape are compared to the parts in $LEVEL EXEC. If a mismatch is found then the level of the service tool parts on the tape (if known) is reported. This will ensure that the user is aware that the levels differ. In this context a match is defined to mean:

  - Tape file 1:

    Parts from the tape must be equal in file name, file type, and file mode to the parts identified in $LEVEL EXEC. File name is checked against the names in column 1 and file type is checked against the columns listing the filetypes for the VM products affected by the PTF.

  - Tape file 2:

    Parts from the tape must exist in $LEVEL EXEC. File name is checked against the names on column 1 and file type is checked against the types in column 3.

- Tape level, against tape inventory

  The service tool parts in files 1 and 2 of the service tape are compared to the parts in $LEVEL MAP. If a mismatch is found, a message indicates that a discrepancy exists within the tape itself. In this context a match is defined to mean:

  - Tape file 1:

    Parts from the tape must be equal in file name, file type, and file mode to the parts identified in $LEVEL MAP. File name is checked against the names in column 1 and and file type is checked against the types in column 3.

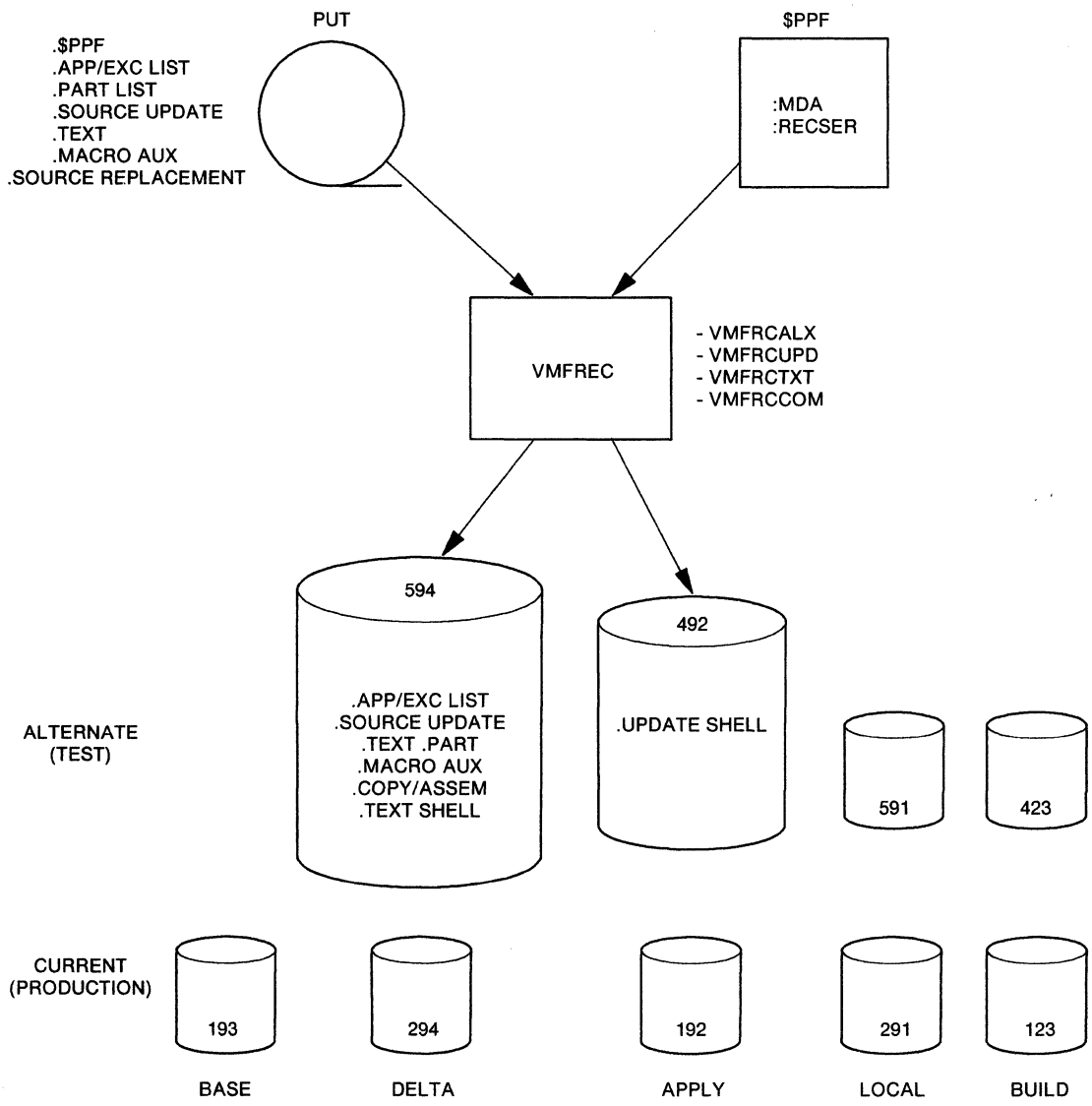
  - Tape file 2:

    Parts from the tape must exist in $LEVEL MAP. File name is checked against the names on column 1 and file type is checked against the types in column 3.

Return Codes

VMFREC issues the following return codes:

Return Code	Explanation
0	No errors were encountered.
4	Minor errors were encountered. Check $VMFREC $ERRLOG.
8	There was a syntax error in the command.
23	Merge processing error.
24	Unexpected tag in PPF.
28	The required file was not found.
100	An unrecoverable error was encountered.



NOTE: The minidisk addresses shown are defaults for CP.

Figure 52 (Part 1 of 2). How VMFREC Works

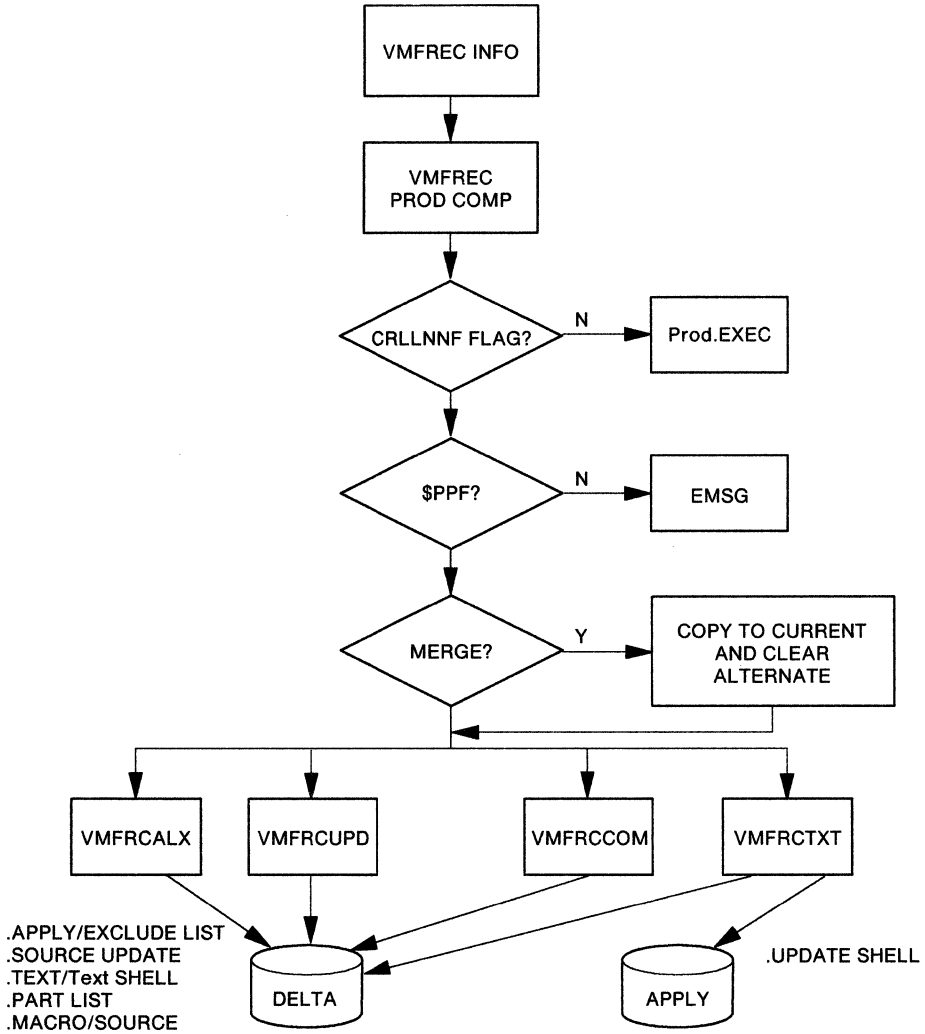


Figure 52 (Part 2 of 2). How VMFREC Works

VMFREMOV EXEC

The VMFREMOV EXEC procedure removes PTFs applied by the VMFMERGE EXEC procedure.

Do not use this procedure to service any of the base components of VM/XA SP. Use this procedure when applying PTFs to System Network Architecture (SNA) products.

Format

The format of the VMFREMOV command is:

VMFREMOV	*prodid*	**PTF** {*ptfnum* | * } **PTFLIST** *remlist* **CONVERT** [*lastfilemode*]

where:

prodid
is the ID of the product.

PTF {*ptfnum* / * }
removes the specified PTF (*ptfnum* is the file name of a PTF) or, if you enter * instead of a ptfnum, removes the PTFs listed in a remove list file named *prodid* REMLIST. This file must already be created and must be on a DELTA disk.

PTFLIST *remlist*
removes the selected PFTs in a remove list file named *remlist* REMLIST. This file must already be created and must be on a DELTA disk.

CONVERT
invokes the tool that creates the reqby log file, if one does not already exist, but does **not** remove any changes.

lastfilemode
specifies the file mode of the last DELTA disk. VMFREMOV assumes that the merge disk is always accessed as E and that all other DELTA disks are accessed as consecutive file modes between F and the *lastfilemode*. If you do not specify *lastfilemode*, then the file modes for the MERGE and DELTA disks are determined from the information in the VMFPARM file. Use the *lastfilemode* parameter only if you know you have the correct disks accessed as the proper modes.

If you wish only to create the reqby log and do **not** want to remove any changes, then use the CONVERT keyword.

How VMFREMOV Works

VMFREMOV is a service process that removes PTFs applied by VMFMERGE. The EXEC procedure:

1. Obtains data from the merge log and the service control file(s) to build the reqby log if one does not already exist. The reqby log contains a list of all dependent PTFs that must be removed if their requisite PTF is removed.
   - If a service control file for any of the PTFs is missing, then processing continues. However, the reqby log will be incomplete if the missing service control file contains requisites. If any SCFs were missing, processing ends after VMFREMOV completes the build of the reqby log. No PTFs are removed.
2. Checks the merge log to insure that the PTF to be removed is currently merged.
3. Reads the reqby log for the list of dependent PTFs to remove.
4. Removes a PTF (for example, `UV00007`) that may supersede other PTFs (for example, `UV00005`). Note that the other PTF (`UV00005`) is no longer superseded and that its status is what it was prior to the merge of the primary PTF (`UV00007`) (that is, merged, superseded, or no status).

   If the prior status of `UV00005` is no status, then VMFREMOV removes its dependents.
5. Copies, from the DELTA disk to the MERGE disk, each element affected by the PTF being removed if previous service for an element is merged. If the element has not been merged, VMFREMOV erases the element from the MERGE disk.

   The very first line of a copied text deck is always a comment line consisting of the PTF name, and the date and time stamp. Any information on the `:apartext` entry is copied, but the first line is a comment.

   Note: A temporary file (with file type of OVMFMGLG) is created during this procedure to insure system integrity. This file is erased during normal VMFREMOV processing.
6. Removes the element's entry from the reqby log. If an element has other elements that are dependent upon it, VMFREMOV also removes those dependent entries from the reqby log.
7. Updates the merge log with the current status of the PTFs. The merged entry is commented out. Another comment is added to the end of the merge log (with a time and date stamp) indicating that the PTF has been removed.

   In the case where a PTF that supersedes another PTF is removed, the SUPERSEDED entry is commented out.

Note: If you remove a change, VMFREMOV has no way of knowing if the change is a requisite of a change in another product. You should have made note of any cross-product requisite information during VMFMERGE processing (see page 696) to know which changes to remove manually.

Messages

DMSWRM002E **File** *fn ft* [*fm*] **not found.**
DMSWRM002W **File** *fn ft* [*fm*] **not found.**
DMSWRM008E **Device** *vdevno* **invalid or nonexistent.**
DMSWRM017E **Invalid device address** *vdevno*.
DMSWRM520E **Invalid operand:** *operand*.
DMSWRM545E **Missing operands.**
DMSWRM632E **I/O error in EXECIO; RC-***nn* **from** *command* **command.**
DMSWRM649E **Extraneous parameter** *parm*.
DMSWRM653E **Error executing** *command*.
DMSWRM823E **PTF** *name1* **is listed as a dependent of PTF** *name2*, **but PTF** *name1* **is not merged.**
DMSWRM824W *prodid* **VMFREQBY may be incomplete due to a missing SCF.**

DMSWRM856E **Disk address** *vdevno* **is listed more than once on the DELTA and/or MERGE entry records in the** *prodid* **VMFPARM file.**
DMSWRM857E **The number of disk addresses on the DELTA entry record cannot exceed nine.**
DMSWRM858E **Unable to find a** *tag* **entry record in the** *fn ft* **file.**
DMSWRM859E **The** *prodid* **VMFPARM file has no disk addresses on the {MERGE|DELTA} entry record.**
DMSWRM860E **Only one {MERGE|DELTA} entry record may appear in the** *prodid* **VMFPARM file.**
DMSWRM861I **Accessing** *disk_type* **disk** *vdevno* **as** *fm.*
DMSWRM862I **Change** *name* **{has been** *action*|**is no longer SUPERSEDED by** *name*}.
DMSWRM863E **The MERGE disk** *vdevno* **must be linked read-write.**
DMSWRM864W **PTF** *name* **will not be** *action* **because it is not** *status.*
DMSWRM865I **Processing PTF** *name.*
DMSWRM866W **No PTFs have been** *action.*
DMSWRM867E **Invalid status** *status* **in** *prodid* **VMFMGLOG for entry** *PTF.*
DMSWRM874E **Invalid entry found at line** *line* **in** *fn ft.*
DMSWRM879W **Change name** *name* **appears more than once in the** *fn ft.*
DMSWRM882E **File** *fn ft* [*fm*] **[from** *name*] **not found on any DELTA disks from the disks from the VMFPARM file.**
DMSWRM883W **PTF** *name* **is not a part of product** *prodid* **and must be** *action* **in product** *prodid.*
DMSWRM888E **Error in** *name* **SCF. No entry for element** *fn ft.*
DMSWRM892E **PTF** *name* **has not been** *action.*
DMSWRM893W **Incomplete processing, not all [required] PTFs were** *action.*

For a complete explanation of each message, refer to *VM/XA SP System Messages and Codes Reference.*

VMFSETUP EXEC

The VMFSETUP EXEC uses the minidisk assignment section of the product parameter file to establish the correct minidisk access order for the component being serviced. It can also be used to release the accesses it established and restore the previous access order. VMFSETUP is invoked by VMFAPPLY, VMFHASM, and VMFBLD. You can also invoke it from the command line.

Format

The format of the VMFSETUP EXEC is:

VMFSETUP	*prodid compname* [PFFTEMP] [(*options* ... [)]] *options:* [ACCESS | RESTORE RETAIN *m m m* ...] [ASM | APP | BLD | REC | ALL]

where:

prodid
is the product number of the product to be serviced. For VM/XA SP, the product identifier is 56643082.

compname
is the name of the component to be serviced. The names for the VM/XA System Product Release 2 components are CP, CMS, GCS, and DV.

PPFTEMP
indicates that the caller has created a $PPFTEMP file that contains only one component (keyword). If not specified, VMFSETUP calls VMFOVER to create a $PPFTEMP file based on the *prodid* $PPF file and any existing override file.

options

ACCESS
indicates that a new minidisk access order should be set up. ACCESS is the default.

Note: If a minidisk is in access mode read-write, and you have specified that minidisk as a read-only disk in your prodcct parameter file, VMFSETUP releases the minidisk and re-accesses it as a read-only disk.

RESTORE
indicates that the accesses should be released and the previous access order restored.

REC
sets up the access order for VMFREC.

APP
sets up the access order for VMFAPPLY. Whenever VMFAPPLY invokes VMSETUP, it specifies APP.

ASM
sets up the access order for VMFHASM. Whenever VMFHASM invokes VMSETUP, it specifies ASM. ASM is the default option.

BLD
sets up the access order for VMFBLD. Whenever VMFBLD invokes VMSETUP, it specifies BLD.

ALL
sets up all the accesses specified in the minidisk assignment section of the product parameter file.

RETAIN *m m m...*
each *m* indicates a file mode that is not to be used by VMFSETUP.

How VMFSETUP Works

If VMFSETUP is invoked from the command line rather than from another EXEC, VMFSETUP invokes VMFOVER.

If the file named *productid* $SETUP exists, the current accesses are validated against the release section of *productid* $SETUP. If the accesses do not match, the user is notified and given the option of terminating or releasing everything but the A-disk.

If the accesses match, the original accesses saved in the restore section of the file with the file name of the product name and a file type of $SETUP are restored. If the RESTORE option was specified, VMFSETUP processing is complete.

If the ACCESS option was specified or chosen by default, VMFSETUP locates the minidisk assignment section (:MDA.) in the chosen component section of the product parameter file. VMFSETUP then saves the current minidisk access order in the restore section of *productid* $SETUP.

Minidisks are accessed according to the process options specified. Table 20 shows the minidisk access order for each option:

Table 20. Minidisk Accesses Established by VMFSETUP EXEC

	REC	APP	ASM	BLD	ALL
TASK	X	X	X	X	X
LOCAL1—LOCAL*n*			X	X	X
APPLY	X	X	X	X	X
DELTA1—DELTA*n*	X	X	X	X	X
LOCAL1—LOCAL*n*		X			
BASE1—BASE*n*	X	X	X	X	X
BUILD1—BUILD*n*	X	X	X	X	X
SYSTEM	X	X	X	X	X

Finally, VMFSETUP does a QUERY SEARCH, displays the results on the screen, and stores the results in the log. In addition, these new accesses are saved in the release section of the file named *productid* $SETUP in the form needed to validate before a subsequent release.

Note: If a minidisk is in access mode read-write, and you have specified that minidisk as a read-only disk in your prodcct parameter file, VMFSETUP releases the minidisk and re-access it as a read-only disk.

VMFVIEW EXEC

The VMFVIEW EXEC invokes XEDIT to allow you to view the exception logs. Using the VMFVIEW EXEC's PF key assignments, you can:

- View all the messages of a specific type
- View all the messages of a specific number
- View the HELP screen for a particular message
- Move backwards and forwards through the displayed exception log.

The PF key assignments that the VMFVIEW EXEC uses are defined in a tailorable profile called VMFVIEW$ PROFILE. See "The VMFVIEW profile file" on page 726 for a description of this profile.

Format

The format of the VMFVIEW EXEC is:

VMFVIEW	**?** *errlog* [**LAST**|**ALL**] *errlog:* [Receive | VMFRec | $VMFRec Apply | VMFApply | $VMFApp Build | VMFBld | $VMFBld | BLD]

where:

?
causes a VMFVIEW help screen to be displayed.

errlog
is the exception log you want to view. Specify:

Receive, **VMFRec**, or **$VMFRec** to view the receive exception log
Apply, **VMFApply**, or **$VMFApp** to view the apply exception log
Build, **VMFBld**, **$VMFBld**, or **BLD** to view the build exception log.

LAST
specifies that only messages from the most recent run in the exception log will be displayed. LAST is the default for VMFVIEW.

ALL
specifies that messages from all runs in the exception log will be displayed.

Default PF Key Assignments

The default PF keys for the VMFVIEW EXEC are:

PF01/13 - Help	Display a HELP screen (see Usage Note 3)
PF02/14 - All	Display ALL messages (see Usage Note 4)
PF03/15 - Quit	Exit VMFVIEW
PF04/16 - Exception	Display CK:, MS:, RQ:, SV:, and WN: messages
PF05/17 - Status	Display ST: messages
PF06/18 - Build	Display BD: messages
PF07/19 - Backward	Display previous screen of messages
PF08/20 - Forward	Display following screen of messages
PF09/21 - OutCompRq	Display RO: messages
PF10/22 - Non-Stat	Display all messages except ST: messages
PF11/23 - Requisite	Display RQ: messages
PF12/24 - Severe	Display SV: messages.

Message Headers

The message headers that appear in the exception logs are:

BD These are "build" messages that give information needed to rebuild the parts affected by the application of service.

CK These are items that you **must** check. These messages contain information that may require user action.

MS These are "mismatch" messages that indicate conflicting control information that **must** be investigated and corrected.

RO These are "requisite out of component" messages. They indicate that PTFs **must** be applied to another component.

RQ These are "requisite" messages that indicate that there are PTFs (in the same component) that are missing and **must** be applied.

ST These are "status" messages. These message give useful information but require no further action.

SV These are "severe" messages. These indicate problems that resulted in the termination of a process. These problems **must** be investigate and corrected.

WN These are "warning" messages that **must** be investigated. They indicate situations that may or may not be a real problem. It is up to the user to decide after investigating the cause of the message.

Usage Notes:

1. PF keys are defined by the file VMFVIEW$ PROFILE. You can change the PF key assignments by editing this file. Instructions for tailoring the PF keys appear at the beginning of this file.
2. The initial set of exception log messages displayed by VMFVIEW is defined in the file VMFVIEW$ PROFILE. The default is the "exception" category of messages.
3. The Help PF key performs one of two functions, depending on the position of the cursor:
   a. When the cursor is on the command line or on a line with no message number, the VMFVIEW help screen will be displayed
   b. When the cursor is on a line that contains a message number, the help screen for the corresponding CMS message number will be displayed. If the requested help screen cannot be found, a message is displayed.
4. The "All" PF key performs one of two functions, depending on the position of the cursor:
   a. When the cursor is on the command line or on a line with no message number, the "All" key causes all of the messages in the log to be displayed
   b. When the cursor is on a line that contains a message number, the "All" key causes all of the messages in the log with this message number to be displayed. Note that these keys are still qualified by the LAST|ALL command line option of VMFVIEW.
5. To gather information about messages, you can issue commands such as FILELIST and XEDIT from the command line.
6. You may find it useful to put a subset of messages into a separate CMS file. To do this you can use the XEDIT PUT command. An example follows:

   If you want to put all BD: (build) messages in a file to process separately, press the appropriate PF key to isolate this subset of messages. (The default is PF06/18.) Then enter the following command on the XEDIT command line:

   ```
   PUT * BUILD
   ```

   This creates a CMS file with the name BUILD $ERRLOG. The PUT command changes the current line on the XEDIT screen. You can enter "TOP" on the XEDIT command line line to make the current line the top line of the file or press another PF key to view a new subset of messages.
7. You may find it useful to add the invocation of VMFVIEW to the service user exits. To do this, include the following commands in your user exit:

   ```
   parse upper arg execname flag retcode
   if function = 'CLEAN-UP' then do
     'EXEC VMFVIEW' execname
   ```
8. If, while viewing messages from the LAST run, you wish to check messages in previous runs, use the "ALL" XEDIT macro. This macro resets the "SET SELECT" and "SET DISPLAY" XEDIT sub-commands that are used by VMFVIEW. To return to a view of only the LAST run, press one of the PF keys that is set to select a subset of messages.

The VMFVIEW profile file

VMFVIEW uses a profile file called VMFVIEW$ PROFILE. You can tailor this file to suit your needs. Figure 53 on page 727 shows the VMFVIEW$ PROFILE file supplied by IBM.

```
************************************************************
* THIS PRODUCT CONTAINS RESTRICTED MATERIALS OF IBM.       *
* COPYRIGHT - 5664-308                                     *
* (C) COPYRIGHT IBM CORPORATION - 1989                     *
*           LICENSED MATERIAL - PROPERTY OF IBM            *
************************************************************
*                                                          *
* Name: VMFVIEW$ PROFILE                                   *
*                                                          *
* Function: Tailorable profile for VMFVIEW.                *
*                                                          *
* Syntax of Entries in the Profile:                        *
*                                                          *
*   * cmt                                                  *
*   VMFVIEW PFnn ( text ) key * cmt                        *
*   VMFVIEW INIT ( text ) key * cmt                        *
*   COMMAND xcmd                                           *
*   MACRO   xmac                                           *
*                                                          *
*   where  cmt   is a comment.                             *
*          nn    is a number between 1 and 24 inclusive.   *
*          text  is the text to describe a PF key.  If     *
*                necessary, this field is truncated to 9   *
*                characters or padded with blanks.         *
*          key   is one of the following:                  *
*                'HELP'     - display help screens         *
*                'ALL'      - display all messages         *
*                'QUIT'     - exit from VMFVIEW            *
*                'BACKWARD' - display previous screen      *
*                'FORWARD'  - display following screen     *
*                'xx:<,yy:<,zz:<...>>> with NO spaces      *
*                           - display messages with the    *
*                             listed headers               *
*                '¬xx:<,¬yy:<,¬zz:<...>>> with NO spaces   *
*                           - display all messages except  *
*                             those with the listed headers*
*          xcmd  is an Xedit command.                      *
*          xmac  is an Xedit macro.                        *
*                                                          *
* Usage Notes:                                             *
*                                                          *
* 1) Use an asterisk '*' to 'comment out' old entries      *
*    instead of deleting them or changing them directly.   *
*    This will make it easier to recreate the original     *
*    definitions.                                          *
*                                                          *
* 2) The valid headers are:                                *
*      BD:, CK:, MS:, RO:, RQ:, ST:, SV:, WN:              *
*                                                          *
* 3) It is recommended that PF keys 13-24 duplicate PF     *
*    keys 1-12 because only PF keys 1-12 are displayed     *
*    in VMFVIEW.  If a PF key is not defined the Xedit     *
*    default is used.                                      *
```

Figure 53 (Part 1 of 2). VMFVIEW$ PROFILE

```
* 4) The entries in this profile are processed in the       *
*    same order in which they appear.  This means that      *
*    you can override previous settings.                    *
*                                                           *
* 5) You may use the Xedit command SET PFxx.  If you do,    *
*    the text describing that PF key will be blank.         *
*                                                           *
* 6) Xedit commands and macros must have their correct      *
*    syntax.                                                *
*                                                           *
* 7) The 3 lines on the top of the screen and the 2 lines   *
*    above the bottom line of the screen are reserved for   *
*    VMFVIEW.                                               *
*                                                           *
*************************************************************
VMFVIEW INIT (Exception) CK:,MS:,RQ:,SV:,WN:
VMFVIEW PF01 (Help     ) HELP
VMFVIEW PF02 (All      ) ALL
VMFVIEW PF03 (Quit     ) QUIT
VMFVIEW PF04 (Exception) CK:,MS:,RQ:,SV:,WN:
VMFVIEW PF05 (Status   ) ST:
VMFVIEW PF06 (Build    ) BD:
VMFVIEW PF07 (Backward ) BACKWARD
VMFVIEW PF08 (Forward  ) FORWARD
VMFVIEW PF09 (OutCompRq) RO:
VMFVIEW PF10 (Non-Stat ) ¬ST:
VMFVIEW PF11 (Requisite) RQ:
VMFVIEW PF12 (Severe   ) SV:
VMFVIEW PF13 (Help     ) HELP
VMFVIEW PF14 (All      ) ALL
VMFVIEW PF15 (Quit     ) QUIT
VMFVIEW PF16 (Exception) CK:,MS:,RQ:,SV:,WN:
VMFVIEW PF17 (Status   ) ST:
VMFVIEW PF18 (Build    ) BD:
VMFVIEW PF19 (Backward ) BACKWARD
VMFVIEW PF20 (Forward  ) FORWARD
VMFVIEW PF21 (OutCompRq) RO:
VMFVIEW PF22 (Non-Stat ) ¬ST:
VMFVIEW PF23 (Requisite) RQ:
VMFVIEW PF24 (Severe   ) SV:
*************************************************************
COMMAND SET ENTER IGNORE COMMAND CURSOR HOME PRIORITY 30
COMMAND SET CMDLINE BOTTOM
COMMAND SET CURLINE ON 4
COMMAND SET MSGLINE ON 2 2 OVERLAY
COMMAND SET MSGMODE ON LONG
COMMAND SET COLOR MSGLINE RED
COMMAND SET COLOR CURLINE WHITE
COMMAND SET COLOR FILEAREA YELLOW
COMMAND SET SCALE OFF
COMMAND SET ETMODE OFF
COMMAND SET SHADOW OFF
COMMAND SET PREFIX OFF
```

Figure 53 (Part 2 of 2). VMFVIEW$ PROFILE

VMFZAP EXEC

Use the VMFZAP EXEC procedure to apply ZAPs and to maintain a record of them in the ZAP log.

Do not use this procedure to service any of the base components of VM/XA System Product. Use this procedure when applying PTFs to System Network Architecture (SNA) products.

This EXEC uses the BASE disk, MERGE disk, and ZAP disk as inputs and produces an updated ZAP disk as output.

Format

The format of the VMFZAP EXEC is:

VMFZAP	*prodid*

where:

prodid
is the ID of the product you want to ZAP.

When you issue VMFZAP, you must specify the name of the product you want to ZAP. VMFZAP accesses the disks using file mode letters E–N. VMFZAP accesses the MERGE disk ahead of the BASE disk. When VMFZAP processing stops, your search hierarchy is restored.

In order to use VMFZAP, you MUST have an A-disk accessed in read/write mode. This disk MUST NOT be the ZAP disk, MERGE disk, or BASE disk. That is, the virtual address of your A-disk MUST NOT appear on the ZAP, MERGE, or BASE records of your VMFPARM file.

How VMFZAP Works

VMFZAP:

1. Uses a parameter file to determine the virtual addresses of the ZAP, MERGE, and BASE disks.
2. Reads the ZAP log and builds a list of TEXT file names with ZAPs applied to them.
   - Erases all TEXT files in this list from the ZAP disk.
   - Applies ZAPs to the first version of the TEXT file found among the other disks.
3. Erases the ZAP log.
4. Reads the merge log and builds a list of ZAPs that are currently superseded.
5. Reads the ZAP List for the names of all ZAPs you want to apply.
6. Checks each ZAP name to see if it is superseded. If it is not superseded, then VMFZAP reads the control file for that ZAP.

   Control files for a product ZAP may contain information for ZAPping more than one text file. VMFZAP separates this information by text file name and processes the ZAP of each text file in the order they are listed in the control file.

   Checks each text file name to see if it resides on some disk other than the ZAP disk. It is an error if the text file resides on the ZAP disk already.

If the text file exists, VMFZAP writes a temporary file called $$VMFZAP ZAP to the ZAP disk containing the ZAP control records for the current text file.

Copies the text file to the ZAP disk under a temporary name of the format VMF$T*n* TEXT, where *n* is a number determined by how many text files are affected by the ZAP currently being processed. When this temporary file has been successfully ZAPped it is renamed to its original name on the ZAP disk.

7. Calls ZAPTEXT passing the $$VMFZAP name and the VMF$T*n* file name to be ZAPped.

   Refer to "ZAPTEXT EXEC" on page 742 for more information about the ZAPTEXT command.

8. Updates the ZAP log with the information about the TEXT file that was just ZAPped.
9. Restores your disk search hierarchy once all ZAPs in the ZAPLIST have been processed.

Messages

DMSWZP002E **File** *fn ft* [*fm*] **not found**
DMSWZP008E **Device** *vdev* **invalid or nonexistent**
DMSWZP017E **Invalid device address** *vdev*
DMSWZP520E **Invalid operand :** *operand*
DMSWZP545E **Missing operands**
DMSWZP649E **Extraneous parameter** *parm*
DMSWZP653E **Error executing** *command*
DMSWZP856E **Disk address** *vdev* **is listed more than once on the {BASE, ZAP | DELTA} and/or MERGE entry records in the** *prodid* **VMFPARM file.**
DMSWZP858E **Unable to find a** *tag* **entry record in the** *fn ft* **file.**
DMSWMG859E **The** *prodid* **VMFPARM file has no disk addresses on the {BASE | MERGE | ZAP} entry record.**
DMSWMG860E **Only one {BASE | MERGE | ZAP} entry record may appear in the** *prodid* **VMFPARM file.**
DMSWZP861I **Accessing** *disk_type* **disk** *vdev* **as** *mode.*
DMSWZP862I **ZAP** *name* **has been** *action*
DMSWZP863E **The ZAP disk** *vdev* **must be linked read-write.**
DMSWZP864W **ZAP** *name* **will not be** *action* **because it already is** *status*
DMSWZP865I **Processing ZAP** *name*
DMSWZP874E **Invalid entry found at line** *line* **in** *fn ft*
DMSWZP875E **File** *fn ft* [*fm*] **not found on any disks from the VMFPARM file.**
DMSWZP876E **The total number of disk addresses on the BASE and MERGE entry records cannot exceed nine.**
DMSWZP877W *fn* **TEXT was previously zapped but was not found on the ZAP disk.**
DMSWZP878E *prodid* **ZAPLIST does not contain any unsuperseded zap names. No zaps will be applied.**
DMSWZP879W **ZAP name** *name* **appears more than once in the** *fn ft* **[It will only be applied once.]**
DMSWZP880E **Error in ZAPTEXT while processing** *fn* **TEXT. Text files affected by** *file name2* **ZAP will not be saved on the ZAP disk.**
DMSWZP881E *fn* **TEXT was found on the zap disk but was not zapped during the VMFZAP run. This file should not be on the ZAP disk.**
DMSWZP885I **File** *prodid* **VMFZPLOG not found on the ZAP disk. No text files will be removed from the ZAP disk.**
DMSWZP886E **File name** *name* **from the** *fn ft* [*fm*] **file is longer than 8 characters.**
DMSWZP887E **Record number** *number* **from the** *fn ft* [*fm*] **file is longer than 80 bytes.**

For a complete explanation of each message, refer to *VM/XA SP System Messages and Codes Reference.*

VSEVSAM EXEC

Use the VSEVSAM EXEC to obtain VSE/VSAM Assembler Language macros from the Licensed Optional Machine Readable Materials tape. The VSEVSAM EXEC creates the VSEVSAM MACLIB for you. Once the MACLIB is created, it contains all of the VSE/VSAM assembler language macros, and the following VSE macros: CDLOAD, CLOSE, CLOSER, GET, OPEN, OPENR, and PUT.

The format of the VSEVSAM EXEC is:

VSEVSAM	

Example of Using VSEVSAM

Before invoking the VSEVSAM EXEC, complete the following:

1. Mount the Licensed Optional Machine Readable Materials tape at virtual address 181.
2. Load the seven VSE macros from the Product Tape to MAINT 393 or a minidisk of your choice. (As long as the macros are available when VSEVSAM is invoked, the actual minidisk used is not critical.)

   Note: Because the seven VSE macros (CDLOAD, CLOSE, CLOSER, GET, OPEN, OPENR, and PUT) will be loaded into the MACLIB, they can be erased from the disk after the MACLIB is created.

To invoke VSEVSAM EXEC, enter:

```
vsevsam ■
```

Once invoked, VSEVSAM EXEC prompts you for information. For example, the system responds:

```
DMSWVV797I "QUIT" may be entered in response to any query to
           end processing.

DMSWVV788R Are the macros to be read from tape or are
           they already on disk?  Reply  TAPE or DISK.
           If a default of TAPE is to be used, press "ENTER."
```

For this example, you read the macros from disk. Enter:

```
disk ■
```

The system responds:

```
DMSWVV790R If the default library name of "VSEVSAM" is to be
           used, press "ENTER."  Else, enter the name to be
           used for the library.
```

You want to call the library VSAMMACS, so enter:

```
vsammacs ■
```

The system responds:

```
DMSWVV791I The library name will be "VSAMMACS."
           Press "ENTER" to continue, else enter
           "QUIT" or the name to be used for the library.
```

Because VSAMMACS is the name we want to call the library, press:

■

The system responds:

```
DMSWVV808R Macro library libname will be erased.
           Press "ENTER" to continue or type "QUIT" to exit.
```

Because we want to erase old versions of the library that might be on our A-disk, press:

■

The system responds:

```
DMSWVV793I Maclib generation completed.

DMSWVV792R Are the macros to be erased from disk?
           Reply (YES | NO).  Press "ENTER" for default
           of "YES."
```

We want to erase the macros from disk, so press:

■

The system responds:

```
DMSWVV802I Macros erased - VSEVSAM processing complete.
Ready;
```

Messages

DMSWVV788R **Are the macros to be read from tape or are they already on disk? Reply TAPE or DISK. If a default of TAPE is to be used, press "ENTER".**

DMSWVV789W **Invalid response.**

DMSWVV790R **If the default library name of "VSEVSAM" is to be used, press "ENTER". Else, enter the name to be used for the library.**

DMSWVV791I **The library name will be *libname*. Press "ENTER" to continue, else enter "QUIT" or the name to be used for the library.**

DMSWVV792R **Are the macros to be erased from disk? Reply (YES | NO). Press "ENTER" for default of "YES".**

DMSWVV793I **Maclib generation completed.**

DMSWVV794E **Error in maclib generation.**

DMSWVV795E **Error reading macros from tape.**

DMSWVV796E **Error reading from "VSEVSAM SCAN" file.**

DMSWVV797I **"QUIT" may be entered in response to any query to end processing.**

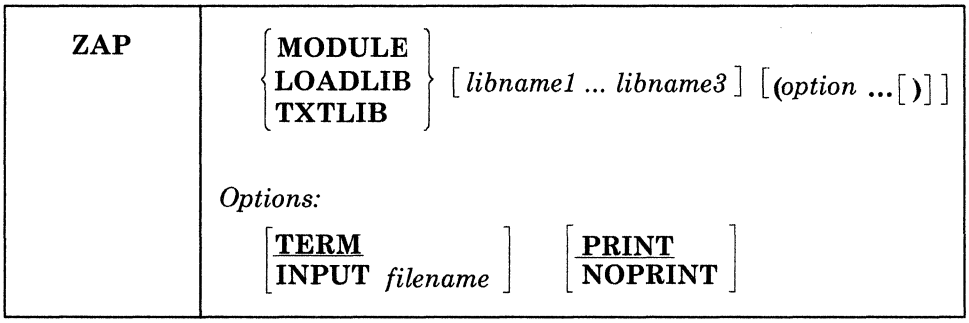
DMSWVV798R	**The VSE/VSAM Optional Source Statement Library tape must be mounted as virtual 181. If it is not, enter "QUIT" here and have the tape mounted. Else, press "ENTER" to continue.**
DMSWVV799E	**Error reading from "VSEVSAM SCAN" file - all macros may not be erased.**
DMSWVV800E	**One of the files needed for maclib generation is missing.**
DMSWVV801I	**Arguments entered are ignored.**
DMSWVV802I	**Macros erased - VSEVSAM processing complete.**
DMSWVV808R	**Macro library *libname* will be erased. Press "ENTER" to continue or type "QUIT" to exit.**
DMSWVV809E	**Error copying "VSEVSAM SCAN" file from S-disk to A-disk.**

ZAP MODULE

ZAP is a CMS command that changes or dumps MODULE, LOADLIB, or TXTLIB files. It may be used to change either fixed or variable length MODULE files. It is for use by system support personnel only.

Input control records control ZAP processing. They can be submitted either from the terminal or from a disk file. Using the VER and REP control records, you can verify and replace data or instructions in a control section (CSECT). Using the DUMP control record, you can dump all or part of a CSECT, an entire member of a LOADLIB or TXTLIB file, or an entire module of a MODULE file.

Format



where:

MODULE
LOADLIB
TXTLIB
indicates the type of file that you want to change or dump.

libname
is the file name of the library containing the member you want to change or dump. You can specify one to three library names. This operand is valid only for LOADLIB and TXTLIB files.

options...

TERM [**PRINT** | **NOPRINT**]

indicates that input to the ZAP service program is submitted through the terminal. If you specify TERM, the prompting message `ENTER:` is issued, and you can then enter input control records up to 80 characters long. If you specify PRINT with TERM, all output prints on the printer, but only error messages display at the terminal. If you specify NOPRINT with TERM, nothing prints on the printer, and all output except control records displays at the terminal. See Table 21 on page 735.

INPUT *filename* [**PRINT** | **NOPRINT**]

specifies that input is submitted from a disk file called *file name*. This file must have a file type of ZAP, and must be a fixed 80-byte sequential file residing on any accessible device. If you specify PRINT with INPUT *file name*, all output produced by the ZAP service program prints on the printer. In addition, commands in error, control records in error, and error messages display at the terminal. If you specify NOPRINT with INPUT *file name*, nothing prints on the printer, and all output displays at the terminal. See Table 21 on page 735.

Table 21. Valid Options and Their Output		
	PRINT	**NOPRINT**
INPUT	Everything prints on the printer. Commands in error, control records in error, and error messages display on the terminal.	Nothing prints on the printer. Everything displays on the terminal.
TERM	Everything prints on the Printer. Error messages display on the terminal.	Nothing prints on the printer. Everything except control records displays on the terminal.

Input Control Records

There are eight types of ZAP control records:

- DUMP
- NAME
- BASE
- VER or VERIFY
- REP
- LOG
- COMMENT
- END.

The ZAP program can accept only 80 characters of data for each control record. ZAP control records are free-form and need not start in position one of the record. Separate all information by one or more blanks. All address fields including disp (displacement) fields in VER and REP control records must contain an even number of hexadecimal digits, to a maximum of six digits (X'OD', X'02C8', X'014318'). Data fields in VER and REP control records must also contain an even number of hexadecimal digits.

If you want, you can separate the data anywhere by commas (for example, 83256482 or 8325,6482). The commas have no effect on the operation.

Note: Do not use blank spaces as separators *within* data fields.

The program sets the NOGO switch on if it finds a control record in error. A file cannot be changed if the NOGO switch is turned on. The next valid NAME record turns the NOGO switch off. This means that if the control record is the NAME record, all succeeding records are ignored until the next NAME, DUMP, or END record. For any other error, only REP control records that follow are ignored.

DUMP Control Record

The DUMP control record allows you to dump a portion or all of a specified control section, or the complete member or module. The format of the output of the dump is hexadecimal with an EBCDIC translation of the hexadecimal data.

The DUMP control record is optional and resets the NOGO switch off. The DUMP control record must not immediately precede a BASE, VER, or REP control record. A NAME control record must precede the BASE, VER, and REP control records (if any) that follow a DUMP control record.

Format:

```
DUMP  { membername  }  { csectname  [ startaddress  [ endaddress ] ] }
      { modulename  }  { ALL                                        }
```

where:

membername
is the name of the member you want to dump, or the member that contains the CSECT(s) you want to dump. This member must be in one of the libraries you specify on the ZAP command.

For a CMS TXTLIB, the format of the dump control record requires that you specify both *membername* and *CSECTname*. Because the library directory does not contain member names, any word may be used to replace *membername*. The program searches for only the second name following the dump operand; therefore, the second name must be *CSECTname*.

modulename
is the name of the module you want to dump, or the module that contains the CSECT(s) you want to dump. If you specify a module that has no loader table, the program dumps the entire module.

CSECTname
is the name of the control section that you want to dump. If you do not specify *CSECTname*, the program dumps only the first CSECT. *CSECTname* is required for CMS TXTLIBs, but is optional for OS TXTLIBs, LOADLIBs, and MODULE files. (See the discussion of CSECTname in "NAME Control Record" on page 737). You must not specify *CSECTname* for a module created with the NOMAP option.

startaddress
is the location within the specified CSECT where you want the dump to begin. This must be two, four, or six hexadecimal digits. The start address is the displacement from the beginning of the CSECT. For example, to start dumping at address 08 in a CSECT that begins at location X'400', you specify start address X'08', not X'0408'.

endaddress
is the last address you want to dump. This must be two, four, or six hexadecimal digits. If you do not specify *endaddress*, the program dumps from *startaddress* to the end of the CSECT. Note that start and end addresses apply only when you specify *CSECTname*.

ALL
tells the program to dump all CSECTs within the member or module you specify. You can specify ALL for MODULE files, LOADLIBs, and OS TXTLIBs, but not for CMS TXTLIBs. To dump all the CSECTs in a member of a CMS TXTLIB, you must issue a separate DUMP control record for each CSECT.

Usage Notes:

1. Displacements listed in the dump output for a module file are calculated from the beginning location of the module. Therefore, the addresses in the output may differ from the displacements within a CSECT.
2. If a DUMP control record references a TXTLIB file that contains ORG statements causing more than one occurrence of an address, data found at the first occurrence is displayed, but any subsequent redefinition of data for the same location is ignored.

NAME Control Record

The NAME control record specifies the member or module and CSECT that contain the data you want the ZAP operation to verify or replace. The NAME control record must precede the BASE, VER, and REP control records. If it does not, the program sets the NOGO switch on.

Format:

```
NAME  { membername }  [csectname]
      { modulename }
```

where:

membername
is the member that you want to search for the desired CSECT.

modulename
is the module that you want to search for the desired CSECT.

CSECTname
is the name of the control section you want to change.

Usage Notes:

1. You must specify *CSECTname* if the CSECT you want to change is in a CMS TXTLIB (that is, a TXTLIB created by the TXTLIB command from CMS TEXT decks that do not have a NAME card following the END card). The directory of a CMS TXTLIB contains only CSECT names and no member names. Select a word to replace *membername* as the first entry following the NAME operand in the NAME statement for a CMS TXTLIB.

   Note: The word you specify for the *membername* for a CMS TXTLIB should be a meaningful name. The file name of the LOG control record is determined by the *membername* or *modulename* you specify in the NAME control record.

2. The CSECT name you specify in the NAME record is compared with CSECT names in the directory. If the CSECT(s) match and no member name match is found, the member selected is the one that contains the CSECT name.

3. *CSECTname* is optional if the CSECT you want to change is a LOADLIB or an OS TXTLIB (that is, a TXTLIB created by the TXTLIB command from CMS TEXT decks that have a NAME card after the END card). The dictionaries of the specified libraries are searched for the member name and the member is then searched for the CSECT name, if you specified one. If you do not specify *CSECTname* for a LOADLIB or an OS TXTLIB, the program uses the first control section.

4. *CSECTname* is optional for a MODULE file. The module named in the NAME control record is found and, if you specify *CSECTname*, the first record is read to determine the number of records in the module and the availability of a loader table, which the program can then search for *CSECTname*. If you do not specify *CSECTname*, the program uses the beginning location of the module. You cannot specify *CSECTname* if the module was created with the NOMAP option.

BASE Control Record

The BASE control record adjusts displacement values for subsequent VER or REP control records for a CSECT whose starting address is not location zero in an assembly listing.

The BASE control record is optional. See the discussion under "VER or VERIFY Control Record" on page 738. If you specify the BASE control record, it must follow the NAME record, but it need not follow the NAME record immediately. For example, you could have the following sequence of control records: NAME, VER, REP, BASE, VER, REP.

Format:

BASE *address*

where:

address
is the starting address of the CSECT. It must be two, four, or six hexadecimal digits.

Usage Note: If you do not specify a CSECTname in the NAME control record, you cannot specify any BASE value other than 00.

Example: For a CSECT starting at location X'400', you specify BASE 0400 in the BASE control record. If a subsequent VER card requests verification of location X'0408', BASE 0400 is subtracted from X'0408', and the program verifies location X'08' in the CSECT. This example applies if you specify TXTLIB, LOADLIB, or MODULE and the module map is present.

However, if no module map is present for a MODULE file (that is, the module was generated with the NOMAP option), then all operations are done as if the BASE address is location X'0'. For example, if you specify a BASE of X'400' and the address you want to look at or change in X'408', then you must specify X'08' and not X'408' in the REP and VER control records. The address in this case is from the start of the module.

VER or VERIFY Control Record

The VER control record requests verification of instructions or data within a CSECT. If the verification fails, the program does not do a subsequent REP operation until it finds another NAME control record.

The VER control record is optional. More than one VER record can follow a single NAME record.

Format:

VERIFY *disp* *data*
VER

where:

disp
is the displacement from the start of the CSECT containing the data to be inspected, if you did not submit a BASE control record for this CSECT. *disp* can also be the actual location of the data to be inspected, if you did submit a BASE control record. *disp* must be two, four, or six hexadecimal digits. This displacement does not have to be aligned on a fullword boundary. If this displacement value is

outside the limits of the CSECT specified by the preceding NAME control record, the VERIFY control record is rejected.

data
is the data against which the data in the CSECT is compared. This must be an even number of hexadecimal digits.

Usage Note: If the VER control statement references data in a TXTLIB file that is later redefined by ORG statements, only the first data definition is verified.

Example: If the location you want to verify is X'3CC', and the CSECT begins at location X'2B0', you can enter:

```
base 02B0 ■
ver 03CC data ■
```

or you can omit the BASE control record, subtract the CSECT start address from the address of the data, and enter:

```
ver 011C data ■
```

REP Control Record

The REP control record changes instructions or data at the specified location within the CSECT that you specify in a preceding NAME control record. The data specified in the REP control record replaces the data at the CSECT location specified by the disp operand. This replacement is on a one-for-one basis; that is, one byte of data defined in the control record replaces one byte of data at the location that you specified. If the replacement fails, the program does not do additional REP operations until it finds another NAME control record.

The REP control record is optional. More than one REP record can follow a single NAME record.

Format:

REP *disp* *data*

where:

disp
is the displacement from the start of the CSECT of the data you want to replace (if you did not submit a BASE control record for this CSECT). *disp* can also be the actual location of the data if you did submit a BASE control record. *disp* must be two, four, or six hexadecimal digits. This displacement need not address a fullword boundary. If this displacement value is outside the limits of the CSECT being changed, the program does not do the replacement operation.

data
is the data that is to replace the data in the CSECT. This must be an even number of hexadecimal digits.

Usage Notes:

1. Although you do not have to verify a location before replacing data, you should do so to make sure that the data being changed is what you expect it to be.
2. If the REP control statement references data in a TXTLIB file that is later redefined by ORG statements, the replacement of data takes place at the first occurrence of the data address of the TXTLIB member.

Example: If the location you want to replace is X'3CC', and the CSECT begins at location X'2B0', you can enter:

```
base 02B0 ■
rep 03CC data ■
```

or you can omit the BASE control record, subtract the CSECT start address from the address of the data, and enter:

```
rep 011C data
```

LOG Control Record

The LOG control record lets you specify, after you apply a fix, a unique fix number, which is recorded in a log file for the module or member. The filename of the log file is the same as *membername* or *modulename* in the NAME control record.

Format:

```
LOG     fixnum    [ filetype [user data] ]
                  [ ZAPLOG               ]
```

where:

fixnum
specifies the number associated with the fix. Its length may vary from one-to-eight alphameric characters.

file type
specifies the file type of the log. The default is ZAPLOG.

user data
specifies any data that you want to enter into the log. If you specify *user data*, you must specify *file type*.

Usage Notes:

1. The LOG control record is optional and is allowed only if valid NAME and REP control records are found. The file name is obtained by the log routine from *modulename* or *membername* in the NAME control record. However, if no LOG control record is found, a dummy log record is written at the end of the user's valid REPs.
2. Log multiple names by including a LOG control record after each name. If the LOG record is not included after each name, error message DMSZAP070E results. Processing continues after the error messages occur.
3. The LOG record is 80 bytes in length and contains the following information:
   - Columns 1-63 contain the fixnum and, if specified, the file type and user data
   - Columns 64-80 contain the date and time of the ZAP.

COMMENT Control Record

The ZAP program ignores COMMENT control records. If the PRINT option is in effect, the program prints the comments.

Format:

* *comment*

There must be at least one blank following the asterisk (*) before you enter the text.

END Control Record

The END control record ends ZAP processing. The END record is required and must be the last control record for input from the console.

Format:

END

Special Considerations for Using the ZAP Service Program

Before using the ZAP command against MODULE files, you can use the MODMAP command to determine whether a module map exists and what it contains.

When a ZAP input file has more than one pair of VER and REP control records, and a VER control record (other than the first) fails, you must remove the records prior to the failing record and correct the error before you issue the ZAP command again. Otherwise, the file being changed returns to its original status.

The REP control record cannot be used to place data in an undefined area such as a Define Storage area. If any part of a data field specified in a pair of VER and REP control records is an undefined area, the system displays warning message DMSZAP248W, and no data replacement occurs. If you do not issue a VER control record prior to the REP control record, some change to data may result. User-defined data may be inserted in undefined areas of text files by using the REP statement described under the LOAD command.

If the file to be dumped contains undefined areas (such as a DS or ORG statement in a TXTLIB member), the hexadecimal portion of the dump contains blanks to indicate that the corresponding positions are undefined.

VER and REP control words can be used to change TXTLIB members produced by FORTRAN compilers that store the length of the compiled text in the END card rather than in the ESD card. However, if a member of this type contains multiple CSECTs, only the first CSECT can be changed by the ZAP program.

The TXT records should be in ascending address order. If ZAP finds a TXT record with an address higher than the specified address, it stops scanning for the specified address. This means that ZAP control records affect only the data at the first occurrence of the address.

When applying ZAPs to a text deck created by a compiler, be aware that some compilers, such as FORTRAN, may generate a text deck in which the TXT records are not in ascending address order.

ZAPTEXT EXEC

ZAPTEXT EXEC modifies or dumps individual text files. Use ZAPTEXT like the ZAP service program, but only for text files, not for MODULEs, TXTLIBs, or LOADLIBs. (Use ZAP to process MODULEs, TXTLIBs, and LOADLIBs.) ZAPTEXT uses the same control information as ZAP and can also use the EXPAND ZAP control word. The user's A-disk must be accessed as read/write to use ZAPTEXT.

Format

The format of the ZAPTEXT EXEC is:

ZAPTEXT	*fn* [*ft* [*fm*]] [(*options:* ... [)]] *Options:* [**INPUT** *filename*] [**PRINT** / **NOPRINT**]

where:

fn ft fm
is the file ID of the text file that you want to change. If you do not specify the file type or file mode, the system assumes a file type of TEXT and a file mode of A1. The file mode must specify a read/write disk.

options...

INPUT *file name*
identifies the file that has the ZAP control records. This file must:

- Have a file type of ZAP, and
- Be a fixed 80-byte sequential file that resides on any accessible disk.

If you do not specify *file name*, it defaults to whatever you specify as *fn* on the ZAPTEXT command.

PRINT
prints all output produced by ZAPTEXT on the printer. The system also displays error messages, commands in error, and control records in error at the terminal.

NOPRINT
does not print anything on the printer, and instead displays all output at the terminal.

ZAPTEXT Input Control Records

Control Records Also Used by ZAP

ZAPTEXT uses the same control information as the current ZAP service program, with the addition of the EXPAND control record. The ZAP service program ignores any EXPAND control records. Refer to "Input Control Records" on page 735 for information about the control records, other than EXPAND, that ZAPTEXT uses.

Use the ZAP control records with ZAPTEXT according to ZAP's TXTLIB conventions.

EXPAND Control Record

The EXPAND control record lets you increase the size of a named control section contained in the text file.

Format: The format of the EXPAND control record is:

```
EXPAND csect size [ , csect size ...]
```

where:

csect
specifies the symbolic name of a control section whose length you want to increase.

size
specifies the decimal number of bytes for the system to add to the control section length. The system initializes the added bytes to binary zero. The maximum number of bytes for each control section that you indicate is 4095.

Control Record Usage Notes

1. Each control record may have multiple entries, but you must separate them with commas. Do not spill an entry onto the next line.
2. The system processes all EXPAND control records before any other control records, regardless of their position in the control file.
3. The effective length of the expansion, which is the actual number of bytes added to the control section, may be greater than the length that you specify for the expansion. This may occur if, after the specified expansion, the system must add padding bytes to align the next control section or common area.
4. When you increase a control section's size, it may affect the offset address of any following control section. This is important when you determine values for BASE, REP, and VER control records. Use the effective expansion lengths when you are determining control section offsets.

EXPAND Command

ZAPTEXT calls EXPAND if you specify an EXPAND control record in the ZAP control file. Use EXPAND to add space to a program in object deck form. The system creates object decks when you assemble or compile a source program. This is especially useful when you do not have the source code for a program or the program does not have a patch area.

Note: EXPAND can add extra space only at the end of named control sections (CSECTs). EXPAND cannot expand private code (unnamed CSECT) and common areas (named or unnamed).

Do not increase the length of a program beyond its design limitations. For example, if you add space to a control section beyond the range of its base register addressability, that space is unusable.

Format

The format of the EXPAND command is:

EXPAND	*fn1* [*ft1* [*fm1* [*fn2* [*ft2* [*fm2*]]]]] [(*options:* ... [ )]] *Options:* [**INPUT** *filename* / **CSECT** *csect* **SIZE** *size*] [**PRINT** / **NOPRINT**]

where:

fn1 ft1 fm1
identifies the input text file that the system expands. The file must have valid object deck information, like that created by an assembler or compiler. If you do not specify the file type or file mode, the system assumes a filetype of TEXT and filemode of A1.

Note: EXPAND assumes that the input text file follows OS/VS standards and that the OS/VS Linkage Editor will accept it without error. The system does a limited check for errors. If the input file is invalid, the system may not expand the text file correctly.

fn2 ft2 fm2
identifies the output text file that the system creates. You can use an equal sign (=) for any of the file identifiers to indicate that it is the same identifier as *fn1*, *ft1*, or *fm1*. The default fileid is *$fn1* = =. The system truncates the file name (*fn1*) to 7 characters before appending the $. In any case, *fm2* must be a read/write disk and cannot be an asterisk (*).

options...

INPUT *file name*
identifies an EXPAND input file that contains EXPAND control records. If you do not specify INPUT, the file name defaults to the name of the text file that you are expanding (*fn1*). The file type must be EXPAND. The system searches all accessed disks for this file.

Do not specify this option with the CSECT or SIZE options.

CSECT
specifies the symbolic name of a control section whose length the system will increase. If you specify CSECT, you must also specify SIZE.

Do not specify CSECT with the INPUT option.

SIZE
specifies the decimal number of bytes that the system adds to the control section length. The system initializes the added bytes to binary zeros. The maximum number of bytes for each control section is 4095. If you specify SIZE, you must also specify CSECT.

Do not specify SIZE with the INPUT option.

PRINT
prints on the printer all output that EXPAND produces. In addition, the system displays error messages, commands in error, and control records in error at the terminal.

NOPRINT
does not print any output on the printer, and instead displays it at the terminal.

Messages

- After the system expands each CSECT that you specified, the system issues a message indicating the following:
  - The number of bytes added to the control section.
  - Whether the number of bytes added is greater than the length that you specify for the expansion. This may occur if, after the specified expansion, the system must add padding bytes to align the next control section or common area.
  - The offset, relative to the start of the specified control section where the expansion began.
- If the system finds an error during processing, it stops the update and does not do the expansions.

Appendix C. VM/XA System Product Starter System Information

This section presents the following information:

- The minimum hardware configuration needed to generate VM/XA System Product.
- Sample files that IBM provides with the VM/XA System Product
  (For a sample product parameter file, see "A Sample Product Parameter File" on page 402.).
- Allocations of the system residence device for starter systems.
- Minidisk maps of 3350, 3375, 3380, 3380-E4, and 3380-K system residence devices, based on sample directories.

Minimum Hardware Configuration

The minimum hardware configuration needed to generate VM/XA System Product is:

- One processor (that supports VM/XA SP) with:
  - Processor in 370-XA mode
  - At least 4 megabytes of real storage
  - One processor unit
  - One processor controller
  - One channel processor.
- One system console
- One printer
- Direct access storage devices. The number required depends on the type of DASD used.

DASD Type	Number of DASD
3350	4
3375	3
3380	2
3380-E4	1
3380-K	1

- One magnetic 9-track tape unit with data density of 6250 BPI
  (A second tape unit is recommended, but not required.)
- One operator's console.

IBM supplies the following VM/XA SP starter systems:

3350
3375
3380 (for 3380, 3380-E4, and 3380-K DASD).

DMSNGP ASSEMBLE (CMS Nucleus Generation Profile)

This profile is part of the Starter System. Within this file, you must change CYLADDR=? to the address of the starting cylinder or block at which the nucleus is to be written. Also, you may want to change the VERSION and INSTID to your own names. Change these to whatever is appropriate for your installation.

```
NGP      TITLE 'DMSNGP     (CMS)      VM/XA SYSTEM PRODUCT 5664-308'
         EJECT
DMSNGP   CSECT
         DEFNUC SYSDISK=190,        * S-disk address                 *
               YDISK=19E,           * Y-disk address                 *
               HELP=19D,            * Help disk address              *
               LANGID=AMENG,        * Default is American English    *
               DBCS=NO,             * Default is not a DBCS lang     *
               LANGLEV=5,           * DCSS ID for multiple DCSS      *
               SAVESYS=NO,          * USING CMS IN DCSS YES OR NO    *
               SYSNAME=CMS,         * Name of above DCSS to save     *
               USEINST=YES,         * Using EXEC/XEDIT in DCSS       *
               INSTSEG=CMSINST,     * Name of above DCSS to save     *
               REWRITE=YES,         * Write nucleus yes or no        *
               IPLADDR=190,         * Address of where to write      *
               CYLADDR=?,           * CYL/BLK OF WHERE TO WRITE      *
               IPLCYL0=YES,         * write ipl text on cyl 0        *
               VERSION=?,           * VM/XA CMS                      *
               INSTID='VM/XA CONVERSATIONAL MONITOR SYSTEM'
         END
```

Sample HCPRIO ASSEMBLE File

IBM supplies the following sample file with VM/XA SP. It is only a sample file, and must be tailored to your installation before it can be used.

```
RIO TITLE 'HCPRIOXA - HCPRIO FOR VM/XA SYSTEM PRODUCT   '
*****************************************************************
***     Virtual Machine / System Product       5664-308    ***
***     Contains restricted materials of IBM               ***
***     Copyright (c) I B M   Corporation          1988    ***
***     Licensed Materials -  Property of I B M            ***
***     Refer to Copyright Instructions: Form G120-2083    ***
*****************************************************************
*****************************************************************
*
* MODULE NAME - HCPRIO
*
* DESCRIPTIVE NAME - DEFINITION OF REAL DEVICES
*
*
* FUNCTION - TO DEFINE THE REAL DEVICES
*
*   REGISTER CONVENTIONS - SYMBOLIC REFERENCES TO REGISTERS ARE
*                          OF THE FORM "RX" WHERE X IS A NUMBER
*                          RANGING FROM 0 TO 15.
*
*   PATCH LABEL - NONE
*
* MODULE TYPE - DATA
*
*   PROCESSOR - ASSEMBLER H (VERSION 2)
*
*   ATTRIBUTES - RESIDENT, DATA-ONLY
*
* MACROS -
*
*          RDEVICE  - DEFINE REAL DEVICES
*          RIOGEN   - END REAL DEVICE GENERATION
*
*
* GENERAL COMMENTS - THIS MODULE IS THE USER'S RESPONSIBILITY.
*                    THIS SAMPLE IS BASED UPON THE STARTER IOCP
*                    INPUT FILE.
*
*********************************************************************
         PRINT NOGEN
         EJECT
*---------------------------------------------------------------
* THE RDEVICE MACROS ARE CODED HERE, ONE FOR EACH DEVICE OR
* GROUP OF DEVICES
*---------------------------------------------------------------
*
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(070,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(170,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(270,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(370,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(470,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(570,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(670,8)
```

```
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(770,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(870,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(970,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(A70,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(B70,8)
*
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(080,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(180,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(280,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(380,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(480,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(580,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(680,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(780,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(880,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(980,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(A80,8)
 RDEVICE DEVTYPE=3420,FEATURE=DUALDENS,MODEL=6,DEVNO=(B80,8)
*
 RDEVICE DEVTYPE=3330,MODEL=1,DEVNO=(078,8)
 RDEVICE DEVTYPE=3330,MODEL=1,DEVNO=(178,8)
 RDEVICE DEVTYPE=3330,MODEL=1,DEVNO=(278,8)
 RDEVICE DEVTYPE=3330,MODEL=1,DEVNO=(378,8)
 RDEVICE DEVTYPE=3330,MODEL=1,DEVNO=(478,8)
 RDEVICE DEVTYPE=3330,MODEL=1,DEVNO=(578,8)
*
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(050,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(150,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(250,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(350,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(450,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(550,8)
*
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(058,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(158,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(258,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(358,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(458,8)
 RDEVICE DEVTYPE=3330,MODEL=11,DEVNO=(558,8)
*
 RDEVICE DEVTYPE=3340,DEVNO=(0C0,8)
 RDEVICE DEVTYPE=3340,DEVNO=(1C0,8)
 RDEVICE DEVTYPE=3340,DEVNO=(2C0,8)
 RDEVICE DEVTYPE=3340,DEVNO=(3C0,8)
 RDEVICE DEVTYPE=3340,DEVNO=(4C0,8)
 RDEVICE DEVTYPE=3340,DEVNO=(5C0,8)
*
 RDEVICE DEVTYPE=3350,DEVNO=(048,8)
 RDEVICE DEVTYPE=3350,DEVNO=(148,8)
 RDEVICE DEVTYPE=3350,DEVNO=(248,8)
 RDEVICE DEVTYPE=3350,DEVNO=(348,8)
 RDEVICE DEVTYPE=3350,DEVNO=(448,8)
 RDEVICE DEVTYPE=3350,DEVNO=(548,8)
*
 RDEVICE DEVTYPE=3350,DEVNO=(030,8)
 RDEVICE DEVTYPE=3350,DEVNO=(130,8)
 RDEVICE DEVTYPE=3350,DEVNO=(230,8)
 RDEVICE DEVTYPE=3350,DEVNO=(330,8)
 RDEVICE DEVTYPE=3350,DEVNO=(430,8)
```

```
 RDEVICE DEVTYPE=3350,DEVNO=(530,8)
*
 RDEVICE DEVTYPE=3350,DEVNO=(060,8)
 RDEVICE DEVTYPE=3350,DEVNO=(160,8)
 RDEVICE DEVTYPE=3350,DEVNO=(260,8)
 RDEVICE DEVTYPE=3350,DEVNO=(360,8)
 RDEVICE DEVTYPE=3350,DEVNO=(460,8)
 RDEVICE DEVTYPE=3350,DEVNO=(560,8)
*
 RDEVICE DEVTYPE=3375,DEVNO=(090,8)
 RDEVICE DEVTYPE=3375,DEVNO=(190,8)
 RDEVICE DEVTYPE=3375,DEVNO=(290,8)
 RDEVICE DEVTYPE=3375,DEVNO=(390,8)
 RDEVICE DEVTYPE=3375,DEVNO=(490,8)
*
 RDEVICE DEVTYPE=3375,DEVNO=(068,8)
 RDEVICE DEVTYPE=3375,DEVNO=(168,8)
 RDEVICE DEVTYPE=3375,DEVNO=(268,8)
 RDEVICE DEVTYPE=3375,DEVNO=(368,8)
 RDEVICE DEVTYPE=3375,DEVNO=(468,8)
 RDEVICE DEVTYPE=3375,DEVNO=(568,8)
 RDEVICE DEVTYPE=3375,DEVNO=(668,8)
*
 RDEVICE DEVTYPE=3380,DEVNO=(038,8)
 RDEVICE DEVTYPE=3380,DEVNO=(138,8)
 RDEVICE DEVTYPE=3380,DEVNO=(238,8)
 RDEVICE DEVTYPE=3380,DEVNO=(338,8)
 RDEVICE DEVTYPE=3380,DEVNO=(438,8)
 RDEVICE DEVTYPE=3380,DEVNO=(538,8)
 RDEVICE DEVTYPE=3380,DEVNO=(638,8)
*
 RDEVICE DEVTYPE=3380,DEVNO=(0A0,8)
 RDEVICE DEVTYPE=3380,DEVNO=(1A0,8)
 RDEVICE DEVTYPE=3380,DEVNO=(2A0,8)
 RDEVICE DEVTYPE=3380,DEVNO=(3A0,8)
 RDEVICE DEVTYPE=3380,DEVNO=(4A0,8)
*
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(0F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(1F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(2F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(3F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(4F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(5F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(6F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(7F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(8F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(9F0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(AF0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(BF0,1)
*
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(0D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(1D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(2D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(3D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(4D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(5D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(6D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(7D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(8D0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(9D0,1)
```

```
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(AD0,1)
 RDEVICE DEVTYPE=2305,MODEL=2,DEVNO=(BD0,1)
*
 RDEVICE DEVTYPE=3800,DEVNO=(018,1)
 RDEVICE DEVTYPE=3800,DEVNO=(118,1)
 RDEVICE DEVTYPE=3800,DEVNO=(218,1)
 RDEVICE DEVTYPE=3800,DEVNO=(318,1)
 RDEVICE DEVTYPE=3800,DEVNO=(418,1)
 RDEVICE DEVTYPE=3800,DEVNO=(518,1)
 RDEVICE DEVTYPE=3800,DEVNO=(618,1)
*
 RDEVICE DEVTYPE=3262,DEVNO=(0A8,1)
 RDEVICE DEVTYPE=3262,DEVNO=(1A8,1)
 RDEVICE DEVTYPE=3262,DEVNO=(2A8,1)
 RDEVICE DEVTYPE=3262,DEVNO=(3A8,1)
 RDEVICE DEVTYPE=3262,DEVNO=(4A8,1)
 RDEVICE DEVTYPE=3262,DEVNO=(5A8,1)
*
 RDEVICE DEVTYPE=4245,DEVNO=(0B0,1)
 RDEVICE DEVTYPE=4245,DEVNO=(1B0,1)
 RDEVICE DEVTYPE=4245,DEVNO=(2B0,1)
 RDEVICE DEVTYPE=4245,DEVNO=(3B0,1)
 RDEVICE DEVTYPE=4245,DEVNO=(4B0,1)
 RDEVICE DEVTYPE=4245,DEVNO=(5B0,1)
*
 RDEVICE DEVTYPE=4248,DEVNO=(0B8,1)
 RDEVICE DEVTYPE=4248,DEVNO=(1B8,1)
 RDEVICE DEVTYPE=4248,DEVNO=(2B8,1)
 RDEVICE DEVTYPE=4248,DEVNO=(3B8,1)
 RDEVICE DEVTYPE=4248,DEVNO=(4B8,1)
 RDEVICE DEVTYPE=4248,DEVNO=(5B8,1)
*
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(020,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(120,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(220,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(320,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(420,8)
*
 RDEVICE DEVTYPE=1403,DEVNO=(00E,1)
 RDEVICE DEVTYPE=1403,DEVNO=(10E,1)
 RDEVICE DEVTYPE=1403,DEVNO=(20E,1)
 RDEVICE DEVTYPE=1403,DEVNO=(30E,1)
 RDEVICE DEVTYPE=1403,DEVNO=(40E,1)
 RDEVICE DEVTYPE=1403,DEVNO=(50E,1)
 RDEVICE DEVTYPE=1403,DEVNO=(60E,1)
*
 RDEVICE DEVTYPE=1403,DEVNO=(00F,1)
 RDEVICE DEVTYPE=1403,DEVNO=(10F,1)
 RDEVICE DEVTYPE=1403,DEVNO=(20F,1)
 RDEVICE DEVTYPE=1403,DEVNO=(30F,1)
 RDEVICE DEVTYPE=1403,DEVNO=(40F,1)
 RDEVICE DEVTYPE=1403,DEVNO=(50F,1)
 RDEVICE DEVTYPE=1403,DEVNO=(60F,1)
*
 RDEVICE DEVTYPE=2540R,DEVNO=(00C,1)
 RDEVICE DEVTYPE=2540R,DEVNO=(10C,1)
 RDEVICE DEVTYPE=2540R,DEVNO=(20C,1)
 RDEVICE DEVTYPE=2540R,DEVNO=(30C,1)
 RDEVICE DEVTYPE=2540R,DEVNO=(40C,1)
 RDEVICE DEVTYPE=2540R,DEVNO=(50C,1)
```

```
 RDEVICE DEVTYPE=2540R,DEVNO=(60C,1)
*
 RDEVICE DEVTYPE=2540P,DEVNO=(00D,1)
 RDEVICE DEVTYPE=2540P,DEVNO=(10D,1)
 RDEVICE DEVTYPE=2540P,DEVNO=(20D,1)
 RDEVICE DEVTYPE=2540P,DEVNO=(30D,1)
 RDEVICE DEVTYPE=2540P,DEVNO=(40D,1)
 RDEVICE DEVTYPE=2540P,DEVNO=(50D,1)
 RDEVICE DEVTYPE=2540P,DEVNO=(60D,1)
*
 RDEVICE DEVTYPE=3211,DEVNO=(002,1)
 RDEVICE DEVTYPE=3211,DEVNO=(102,1)
 RDEVICE DEVTYPE=3211,DEVNO=(202,1)
 RDEVICE DEVTYPE=3211,DEVNO=(302,1)
 RDEVICE DEVTYPE=3211,DEVNO=(402,1)
*
 RDEVICE DEVTYPE=3203,MODEL=5,DEVNO=(003,1)
 RDEVICE DEVTYPE=3203,MODEL=5,DEVNO=(103,1)
 RDEVICE DEVTYPE=3203,MODEL=5,DEVNO=(203,1)
 RDEVICE DEVTYPE=3203,MODEL=5,DEVNO=(303,1)
*
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(040,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(140,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(240,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(340,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(440,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(540,8)
 RDEVICE DEVTYPE=3278,MODEL=2,DEVNO=(640,8)
*
 RDEVICE DEVTYPE=3278,MODEL=5,DEVNO=(088,8)
 RDEVICE DEVTYPE=3278,MODEL=5,DEVNO=(188,8)
 RDEVICE DEVTYPE=3278,MODEL=5,DEVNO=(288,8)
 RDEVICE DEVTYPE=3278,MODEL=5,DEVNO=(388,8)
 RDEVICE DEVTYPE=3278,MODEL=5,DEVNO=(488,8)
 RDEVICE DEVTYPE=3278,MODEL=5,DEVNO=(588,8)
*
 RDEVICE DEVTYPE=3290,DEVNO=(098,8)
 RDEVICE DEVTYPE=3290,DEVNO=(198,8)
 RDEVICE DEVTYPE=3290,DEVNO=(298,8)
 RDEVICE DEVTYPE=3290,DEVNO=(398,8)
 RDEVICE DEVTYPE=3290,DEVNO=(498,8)
 RDEVICE DEVTYPE=3290,DEVNO=(598,8)
*
 RDEVICE DEVTYPE=3505,CLASS=A,DEVNO=(012,1)
 RDEVICE DEVTYPE=3505,CLASS=A,DEVNO=(112,1)
 RDEVICE DEVTYPE=3505,CLASS=A,DEVNO=(212,1)
 RDEVICE DEVTYPE=3505,CLASS=A,DEVNO=(312,1)
 RDEVICE DEVTYPE=3505,CLASS=A,DEVNO=(412,1)
*
 RDEVICE DEVTYPE=3525,CLASS=A,DEVNO=(013,1)
 RDEVICE DEVTYPE=3525,CLASS=A,DEVNO=(113,1)
 RDEVICE DEVTYPE=3525,CLASS=A,DEVNO=(213,1)
 RDEVICE DEVTYPE=3525,CLASS=A,DEVNO=(313,1)
 RDEVICE DEVTYPE=3525,CLASS=A,DEVNO=(413,1)
*
 RDEVICE DEVTYPE=3480,DEVNO=(004,4)
 RDEVICE DEVTYPE=3480,DEVNO=(104,4)
 RDEVICE DEVTYPE=3480,DEVNO=(204,4)
 RDEVICE DEVTYPE=3480,DEVNO=(304,4)
*
```

```
 RDEVICE DEVTYPE=3890,DEVNO=(008,4)
 RDEVICE DEVTYPE=3890,DEVNO=(108,4)
*
 RDEVICE DEVTYPE=HFGD,DEVNO=(014,4)
 RDEVICE DEVTYPE=HFGD,DEVNO=(114,4)
*
  EJECT
*************************************************************
*-----------------------------------------------------------
* THE RIOGEN MACRO IS CODED HERE
*-----------------------------------------------------------
         SPACE 2
         RIOGEN CONS=020,ALTCONS=(120,220,320,                  *
               420,040,140,240,340,440,540,640)
```

Sample Input/Output Configuration Profile

IBM supplies the following sample file with VM/XA SP. It is only a sample file, and must be tailored to your installation before it can be used.

```
************************************************************
***     Virtual Machine / System Product       5664-308    ***
***     Contains restricted materials of IBM               ***
***     Copyright (c) I B M   Corporation          1988    ***
***     Licensed Materials -  Property of I B M            ***
***     Refer to Copyright Instructions: Form G120-2083    ***
************************************************************
 ID MSG1='SAMPLE IOCP INPUT FILE',                                    *
               MSG2='COMPANION TO SAMPLE HCPRIO'
 CHPID PATH=((00,0,0),(01,1,0),(02,2,0),(03,3,0),(04,4,0),(05,5,0),   *
               (06,6,0),(07,7,0)),TYPE=BL
 CHPID PATH=((10,8,0),(11,9,0),(12,A,0),(13,B,0)),TYPE=BL
 CHPID PATH=((14,0,1),(15,1,1),(16,2,1),(17,3,1),(20,4,1),(21,5,1),   *
               (22,6,1),(23,7,1)),TYPE=BL
 CHPID PATH=((24,8,1),(25,9,1),(26,A,1),(27,B,1)),TYPE=BL
*IOCP
 CNTLUNIT CUNUMBR=001,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(00,14),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=002,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(01,15),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=003,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(02,16),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=004,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(03,17),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=005,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(04,20),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=006,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(05,21),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=007,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(06,22),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=008,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(07,23),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=009,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(10,24),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=00A,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(11,25),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=00B,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(12,26),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=00C,UNIT=3803,UNITADD=((70,8)),                     *
               PATH=(13,27),PROTOCL=D,SHARED=Y
*IOCP
 CNTLUNIT CUNUMBR=00D,UNIT=3803,UNITADD=((80,8)),                     *
               PATH=(00,14),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=00E,UNIT=3803,UNITADD=((80,8)),                     *
               PATH=(01,15),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=00F,UNIT=3803,UNITADD=((80,8)),                     *
               PATH=(02,16),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=010,UNIT=3803,UNITADD=((80,8)),                     *
               PATH=(03,17),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=011,UNIT=3803,UNITADD=((80,8)),                     *
               PATH=(04,20),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=012,UNIT=3803,UNITADD=((80,8)),                     *
               PATH=(05,21),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=013,UNIT=3803,UNITADD=((80,8)),                     *
```

```
              PATH=(06,22),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=014,UNIT=3803,UNITADD=((80,8)),                *
              PATH=(07,23),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=015,UNIT=3803,UNITADD=((80,8)),                *
              PATH=(10,24),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=016,UNIT=3803,UNITADD=((80,8)),                *
              PATH=(11,25),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=017,UNIT=3803,UNITADD=((80,8)),                *
              PATH=(12,26),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=018,UNIT=3803,UNITADD=((80,8)),                *
              PATH=(13,27),PROTOCL=D,SHARED=Y
*
 CNTLUNIT CUNUMBR=019,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=01A,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=01B,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=01C,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=01D,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=01E,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=01F,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(06,22),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=020,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=021,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=022,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=023,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=024,UNIT=3830,UNITADD=((50,8)),                *
              PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=025,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=026,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=027,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=028,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=029,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=02A,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=02B,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(06,22),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=02C,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=02D,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=02E,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=02F,UNIT=3830,UNITADD=((58,8)),                *
              PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=030,UNIT=3830,UNITADD=((58,8)),                *
```

```
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=031,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=032,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=033,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=034,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=035,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=036,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=037,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(06,22),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=038,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=039,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=03A,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=03B,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=03C,UNIT=3830,UNITADD=((C0,8)),                    *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=03D,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=03E,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=03F,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=040,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=041,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=042,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=043,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(06,22),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=044,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=045,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=046,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=047,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=048,UNIT=3830,UNITADD=((48,8)),                    *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=049,UNIT=3830,UNITADD=((30,8)),                    *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=04A,UNIT=3830,UNITADD=((30,8)),                    *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=04B,UNIT=3830,UNITADD=((30,8)),                    *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=04C,UNIT=3830,UNITADD=((30,8)),                    *
               PATH=(03,17),PROTOCL=D,SHARED=N
```

```
 CNTLUNIT CUNUMBR=04D,UNIT=3830,UNITADD=((30,8)),                *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=04E,UNIT=3830,UNITADD=((30,8)),                *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=04F,UNIT=3830,UNITADD=((30,8)),                *
               PATH=(06,22),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=050,UNIT=3830,UNITADD=((30,8)),                *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=051,UNIT=3830,UNITADD=((30,8)),                *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=052,UNIT=3830,UNITADD=((30,8)),                *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=053,UNIT=3830,UNITADD=((30,8)),                *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=054,UNIT=3830,UNITADD=((30,8)),                *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=055,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=056,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=057,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=058,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=059,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=05A,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=05B,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(06,22),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=05C,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=05D,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=05E,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=05F,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=060,UNIT=3880,UNITADD=((60,8)),                *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=061,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=062,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=063,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=064,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=065,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=066,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=067,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(06,22),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=068,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=069,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(10,24),PROTOCL=D,SHARED=N
```

```
 CNTLUNIT CUNUMBR=06A,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=06B,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=06C,UNIT=2835,UNITADD=((F0,8)),                *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=06D,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=06E,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=06F,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=070,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=071,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=072,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=073,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(06,22),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=074,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=075,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=076,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=077,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=078,UNIT=2835,UNITADD=((D0,8)),                *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=079,UNIT=3800,UNITADD=((18,1)),                *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=07A,UNIT=3800,UNITADD=((18,1)),                *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=07B,UNIT=3800,UNITADD=((18,1)),                *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=07C,UNIT=3800,UNITADD=((18,1)),                *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=07D,UNIT=3800,UNITADD=((18,1)),                *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=07E,UNIT=3800,UNITADD=((18,1)),                *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=07F,UNIT=3800,UNITADD=((18,1)),                *
               PATH=(06,22),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=080,UNIT=3274,UNITADD=((20,8)),                *
               PATH=(07,23),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=081,UNIT=3274,UNITADD=((20,8)),                *
               PATH=(10,24),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=082,UNIT=3274,UNITADD=((20,8)),                *
               PATH=(11,25),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=083,UNIT=3274,UNITADD=((20,8)),                *
               PATH=(12,26),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=084,UNIT=3274,UNITADD=((20,8)),                *
               PATH=(13,27),PROTOCL=D,SHARED=Y
*
 CNTLUNIT CUNUMBR=085,UNIT=2821,UNITADD=((0C,4)),                *
               PATH=(00,14),PROTOCL=D,SHARED=N
```

```
 CNTLUNIT CUNUMBR=086,UNIT=2821,UNITADD=((0C,4)),                    *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=087,UNIT=2821,UNITADD=((0C,4)),                    *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=088,UNIT=2821,UNITADD=((0C,4)),                    *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=089,UNIT=2821,UNITADD=((0C,4)),                    *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=08A,UNIT=2821,UNITADD=((0C,4)),                    *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=08B,UNIT=2821,UNITADD=((0C,4)),                    *
               PATH=(06,22),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=08C,UNIT=3811,UNITADD=((02,1)),                    *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=08D,UNIT=3811,UNITADD=((02,1)),                    *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=08E,UNIT=3811,UNITADD=((02,1)),                    *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=08F,UNIT=3811,UNITADD=((02,1)),                    *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=090,UNIT=3811,UNITADD=((02,1)),                    *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=091,UNIT=3274,UNITADD=((40,8)),                    *
               PATH=(00,14),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=092,UNIT=3274,UNITADD=((40,8)),                    *
               PATH=(01,15),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=093,UNIT=3274,UNITADD=((40,8)),                    *
               PATH=(02,16),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=094,UNIT=3274,UNITADD=((40,8)),                    *
               PATH=(03,17),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=095,UNIT=3274,UNITADD=((40,8)),                    *
               PATH=(04,20),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=096,UNIT=3274,UNITADD=((40,8)),                    *
               PATH=(05,21),PROTOCL=D,SHARED=Y
 CNTLUNIT CUNUMBR=097,UNIT=3274,UNITADD=((40,8)),                    *
               PATH=(06,22),PROTOCL=D,SHARED=Y
*
 CNTLUNIT CUNUMBR=098,UNIT=3505,UNITADD=((12,2)),                    *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=099,UNIT=3505,UNITADD=((12,2)),                    *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=09A,UNIT=3505,UNITADD=((12,2)),                    *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=09B,UNIT=3505,UNITADD=((12,2)),                    *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=09C,UNIT=3505,UNITADD=((12,2)),                    *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=09D,UNIT=3880,UNITADD=((68,8)),                    *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=09E,UNIT=3880,UNITADD=((68,8)),                    *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=09F,UNIT=3880,UNITADD=((68,8)),                    *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0A0,UNIT=3880,UNITADD=((68,8)),                    *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0A1,UNIT=3880,UNITADD=((68,8)),                    *
               PATH=(04,20),PROTOCL=D,SHARED=N
```

```
 CNTLUNIT CUNUMBR=0A2,UNIT=3880,UNITADD=((68,8)),                *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0A3,UNIT=3880,UNITADD=((68,8)),                *
               PATH=(06,22),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=0A4,UNIT=3880,UNITADD=((90,8)),                *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0A5,UNIT=3880,UNITADD=((90,8)),                *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0A6,UNIT=3880,UNITADD=((90,8)),                *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0A7,UNIT=3880,UNITADD=((90,8)),                *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0A8,UNIT=3880,UNITADD=((90,8)),                *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=0A9,UNIT=3880,UNITADD=((38,8)),                *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0AA,UNIT=3880,UNITADD=((38,8)),                *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0AB,UNIT=3880,UNITADD=((38,8)),                *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0AC,UNIT=3880,UNITADD=((38,8)),                *
               PATH=(03,17),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0AD,UNIT=3880,UNITADD=((38,8)),                *
               PATH=(04,20),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0AE,UNIT=3880,UNITADD=((38,8)),                *
               PATH=(05,21),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0AF,UNIT=3880,UNITADD=((38,8)),                *
               PATH=(06,22),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=0B0,UNIT=3880,UNITADD=((A0,8)),                *
               PATH=(07,23),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0B1,UNIT=3880,UNITADD=((A0,8)),                *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0B2,UNIT=3880,UNITADD=((A0,8)),                *
               PATH=(11,25),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0B3,UNIT=3880,UNITADD=((A0,8)),                *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0B4,UNIT=3880,UNITADD=((A0,8)),                *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=0B5,UNIT=3203,UNITADD=((03,1)),                *
               PATH=(00,14),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0B6,UNIT=3203,UNITADD=((03,1)),                *
               PATH=(01,15),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0B7,UNIT=3203,UNITADD=((03,1)),                *
               PATH=(02,16),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0B8,UNIT=3203,UNITADD=((03,1)),                *
               PATH=(03,17),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=0B9,UNIT=3480,UNITADD=((04,4)),                *
               PATH=(04,20),PROTOCL=S,SHARED=N
 CNTLUNIT CUNUMBR=0BA,UNIT=3480,UNITADD=((04,4)),                *
               PATH=(05,21),PROTOCL=S,SHARED=N
 CNTLUNIT CUNUMBR=0BB,UNIT=3480,UNITADD=((04,4)),                *
               PATH=(06,22),PROTOCL=S,SHARED=N
 CNTLUNIT CUNUMBR=0BC,UNIT=3480,UNITADD=((04,4)),                *
               PATH=(07,23),PROTOCL=S,SHARED=N
*
```

```
 CNTLUNIT CUNUMBR=0BD,UNIT=3890,UNITADD=((08,4)),                *
               PATH=(10,24),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0BE,UNIT=3890,UNITADD=((08,4)),                *
               PATH=(11,25),PROTOCL=D,SHARED=N
*
 CNTLUNIT CUNUMBR=0BF,UNIT=5088,UNITADD=((14,4)),                *
               PATH=(12,26),PROTOCL=D,SHARED=N
 CNTLUNIT CUNUMBR=0C0,UNIT=5088,UNITADD=((14,4)),                *
               PATH=(13,27),PROTOCL=D,SHARED=N
*
 IODEVICE CUNUMBR=001,UNIT=3420,ADDRESS=(070,8)
 IODEVICE CUNUMBR=002,UNIT=3420,ADDRESS=(170,8)
 IODEVICE CUNUMBR=003,UNIT=3420,ADDRESS=(270,8)
 IODEVICE CUNUMBR=004,UNIT=3420,ADDRESS=(370,8)
 IODEVICE CUNUMBR=005,UNIT=3420,ADDRESS=(470,8)
 IODEVICE CUNUMBR=006,UNIT=3420,ADDRESS=(570,8)
 IODEVICE CUNUMBR=007,UNIT=3420,ADDRESS=(670,8)
 IODEVICE CUNUMBR=008,UNIT=3420,ADDRESS=(770,8)
 IODEVICE CUNUMBR=009,UNIT=3420,ADDRESS=(870,8)
 IODEVICE CUNUMBR=00A,UNIT=3420,ADDRESS=(970,8)
 IODEVICE CUNUMBR=00B,UNIT=3420,ADDRESS=(A70,8)
 IODEVICE CUNUMBR=00C,UNIT=3420,ADDRESS=(B70,8)
*
 IODEVICE CUNUMBR=00D,UNIT=3420,ADDRESS=(080,8)
 IODEVICE CUNUMBR=00E,UNIT=3420,ADDRESS=(180,8)
 IODEVICE CUNUMBR=00F,UNIT=3420,ADDRESS=(280,8)
 IODEVICE CUNUMBR=010,UNIT=3420,ADDRESS=(380,8)
 IODEVICE CUNUMBR=011,UNIT=3420,ADDRESS=(480,8)
 IODEVICE CUNUMBR=012,UNIT=3420,ADDRESS=(580,8)
 IODEVICE CUNUMBR=013,UNIT=3420,ADDRESS=(680,8)
 IODEVICE CUNUMBR=014,UNIT=3420,ADDRESS=(780,8)
 IODEVICE CUNUMBR=015,UNIT=3420,ADDRESS=(880,8)
 IODEVICE CUNUMBR=016,UNIT=3420,ADDRESS=(980,8)
 IODEVICE CUNUMBR=017,UNIT=3420,ADDRESS=(A80,8)
 IODEVICE CUNUMBR=018,UNIT=3420,ADDRESS=(B80,8)
*
 IODEVICE CUNUMBR=019,UNIT=3330,ADDRESS=(050,8)
 IODEVICE CUNUMBR=01A,UNIT=3330,ADDRESS=(150,8)
 IODEVICE CUNUMBR=01B,UNIT=3330,ADDRESS=(250,8)
 IODEVICE CUNUMBR=01C,UNIT=3330,ADDRESS=(350,8)
 IODEVICE CUNUMBR=01D,UNIT=3330,ADDRESS=(450,8)
 IODEVICE CUNUMBR=01E,UNIT=3330,ADDRESS=(550,8)
 IODEVICE CUNUMBR=01F,UNIT=3330,ADDRESS=(650,8)
 IODEVICE CUNUMBR=020,UNIT=3330,ADDRESS=(750,8)
 IODEVICE CUNUMBR=021,UNIT=3330,ADDRESS=(850,8)
 IODEVICE CUNUMBR=022,UNIT=3330,ADDRESS=(950,8)
 IODEVICE CUNUMBR=023,UNIT=3330,ADDRESS=(A50,8)
 IODEVICE CUNUMBR=024,UNIT=3330,ADDRESS=(B50,8)
*
 IODEVICE CUNUMBR=025,UNIT=3330,ADDRESS=(058,8)
 IODEVICE CUNUMBR=026,UNIT=3330,ADDRESS=(158,8)
 IODEVICE CUNUMBR=027,UNIT=3330,ADDRESS=(258,8)
 IODEVICE CUNUMBR=028,UNIT=3330,ADDRESS=(358,8)
 IODEVICE CUNUMBR=029,UNIT=3330,ADDRESS=(458,8)
 IODEVICE CUNUMBR=02A,UNIT=3330,ADDRESS=(558,8)
 IODEVICE CUNUMBR=02B,UNIT=3330,ADDRESS=(658,8)
 IODEVICE CUNUMBR=02C,UNIT=3330,ADDRESS=(758,8)
 IODEVICE CUNUMBR=02D,UNIT=3330,ADDRESS=(858,8)
 IODEVICE CUNUMBR=02E,UNIT=3330,ADDRESS=(958,8)
 IODEVICE CUNUMBR=02F,UNIT=3330,ADDRESS=(A58,8)
```

```
 IODEVICE CUNUMBR=030,UNIT=3330,ADDRESS=(B58,8)
*
 IODEVICE CUNUMBR=031,UNIT=3340,ADDRESS=(0C0,8)
 IODEVICE CUNUMBR=032,UNIT=3340,ADDRESS=(1C0,8)
 IODEVICE CUNUMBR=033,UNIT=3340,ADDRESS=(2C0,8)
 IODEVICE CUNUMBR=034,UNIT=3340,ADDRESS=(3C0,8)
 IODEVICE CUNUMBR=035,UNIT=3340,ADDRESS=(4C0,8)
 IODEVICE CUNUMBR=036,UNIT=3340,ADDRESS=(5C0,8)
 IODEVICE CUNUMBR=037,UNIT=3340,ADDRESS=(6C0,8)
 IODEVICE CUNUMBR=038,UNIT=3340,ADDRESS=(7C0,8)
 IODEVICE CUNUMBR=039,UNIT=3340,ADDRESS=(8C0,8)
 IODEVICE CUNUMBR=03A,UNIT=3340,ADDRESS=(9C0,8)
 IODEVICE CUNUMBR=03B,UNIT=3340,ADDRESS=(AC0,8)
 IODEVICE CUNUMBR=03C,UNIT=3340,ADDRESS=(BC0,8)
*
 IODEVICE CUNUMBR=03D,UNIT=3350,ADDRESS=(048,8)
 IODEVICE CUNUMBR=03E,UNIT=3350,ADDRESS=(148,8)
 IODEVICE CUNUMBR=03F,UNIT=3350,ADDRESS=(248,8)
 IODEVICE CUNUMBR=040,UNIT=3350,ADDRESS=(348,8)
 IODEVICE CUNUMBR=041,UNIT=3350,ADDRESS=(448,8)
 IODEVICE CUNUMBR=042,UNIT=3350,ADDRESS=(548,8)
 IODEVICE CUNUMBR=043,UNIT=3350,ADDRESS=(648,8)
 IODEVICE CUNUMBR=044,UNIT=3350,ADDRESS=(748,8)
 IODEVICE CUNUMBR=045,UNIT=3350,ADDRESS=(848,8)
 IODEVICE CUNUMBR=046,UNIT=3350,ADDRESS=(948,8)
 IODEVICE CUNUMBR=047,UNIT=3350,ADDRESS=(A48,8)
 IODEVICE CUNUMBR=048,UNIT=3350,ADDRESS=(B48,8)
*
 IODEVICE CUNUMBR=049,UNIT=3350,ADDRESS=(030,8)
 IODEVICE CUNUMBR=04A,UNIT=3350,ADDRESS=(130,8)
 IODEVICE CUNUMBR=04B,UNIT=3350,ADDRESS=(230,8)
 IODEVICE CUNUMBR=04C,UNIT=3350,ADDRESS=(330,8)
 IODEVICE CUNUMBR=04D,UNIT=3350,ADDRESS=(430,8)
 IODEVICE CUNUMBR=04E,UNIT=3350,ADDRESS=(530,8)
 IODEVICE CUNUMBR=04F,UNIT=3350,ADDRESS=(630,8)
 IODEVICE CUNUMBR=050,UNIT=3350,ADDRESS=(730,8)
 IODEVICE CUNUMBR=051,UNIT=3350,ADDRESS=(830,8)
 IODEVICE CUNUMBR=052,UNIT=3350,ADDRESS=(930,8)
 IODEVICE CUNUMBR=053,UNIT=3350,ADDRESS=(A30,8)
 IODEVICE CUNUMBR=054,UNIT=3350,ADDRESS=(B30,8)
*
 IODEVICE CUNUMBR=055,UNIT=3350,ADDRESS=(060,8)
 IODEVICE CUNUMBR=056,UNIT=3350,ADDRESS=(160,8)
 IODEVICE CUNUMBR=057,UNIT=3350,ADDRESS=(260,8)
 IODEVICE CUNUMBR=058,UNIT=3350,ADDRESS=(360,8)
 IODEVICE CUNUMBR=059,UNIT=3350,ADDRESS=(460,8)
 IODEVICE CUNUMBR=05A,UNIT=3350,ADDRESS=(560,8)
 IODEVICE CUNUMBR=05B,UNIT=3350,ADDRESS=(660,8)
 IODEVICE CUNUMBR=05C,UNIT=3350,ADDRESS=(760,8)
 IODEVICE CUNUMBR=05D,UNIT=3350,ADDRESS=(860,8)
 IODEVICE CUNUMBR=05E,UNIT=3350,ADDRESS=(960,8)
 IODEVICE CUNUMBR=05F,UNIT=3350,ADDRESS=(A60,8)
 IODEVICE CUNUMBR=060,UNIT=3350,ADDRESS=(B60,8)
*
 IODEVICE CUNUMBR=061,UNIT=2305,ADDRESS=(0F0,8)
 IODEVICE CUNUMBR=062,UNIT=2305,ADDRESS=(1F0,8)
 IODEVICE CUNUMBR=063,UNIT=2305,ADDRESS=(2F0,8)
 IODEVICE CUNUMBR=064,UNIT=2305,ADDRESS=(3F0,8)
 IODEVICE CUNUMBR=065,UNIT=2305,ADDRESS=(4F0,8)
 IODEVICE CUNUMBR=066,UNIT=2305,ADDRESS=(5F0,8)
```

```
 IODEVICE CUNUMBR=067,UNIT=2305,ADDRESS=(6F0,8)
 IODEVICE CUNUMBR=068,UNIT=2305,ADDRESS=(7F0,8)
 IODEVICE CUNUMBR=069,UNIT=2305,ADDRESS=(8F0,8)
 IODEVICE CUNUMBR=06A,UNIT=2305,ADDRESS=(9F0,8)
 IODEVICE CUNUMBR=06B,UNIT=2305,ADDRESS=(AF0,8)
 IODEVICE CUNUMBR=06C,UNIT=2305,ADDRESS=(BF0,8)
*
 IODEVICE CUNUMBR=06D,UNIT=2305,ADDRESS=(0D0,8)
 IODEVICE CUNUMBR=06E,UNIT=2305,ADDRESS=(1D0,8)
 IODEVICE CUNUMBR=06F,UNIT=2305,ADDRESS=(2D0,8)
 IODEVICE CUNUMBR=070,UNIT=2305,ADDRESS=(3D0,8)
 IODEVICE CUNUMBR=071,UNIT=2305,ADDRESS=(4D0,8)
 IODEVICE CUNUMBR=072,UNIT=2305,ADDRESS=(5D0,8)
 IODEVICE CUNUMBR=073,UNIT=2305,ADDRESS=(6D0,8)
 IODEVICE CUNUMBR=074,UNIT=2305,ADDRESS=(7D0,8)
 IODEVICE CUNUMBR=075,UNIT=2305,ADDRESS=(8D0,8)
 IODEVICE CUNUMBR=076,UNIT=2305,ADDRESS=(9D0,8)
 IODEVICE CUNUMBR=077,UNIT=2305,ADDRESS=(AD0,8)
 IODEVICE CUNUMBR=078,UNIT=2305,ADDRESS=(BD0,8)
*
 IODEVICE CUNUMBR=079,UNIT=3800,ADDRESS=(018,1)
 IODEVICE CUNUMBR=07A,UNIT=3800,ADDRESS=(118,1)
 IODEVICE CUNUMBR=07B,UNIT=3800,ADDRESS=(218,1)
 IODEVICE CUNUMBR=07C,UNIT=3800,ADDRESS=(318,1)
 IODEVICE CUNUMBR=07D,UNIT=3800,ADDRESS=(418,1)
 IODEVICE CUNUMBR=07E,UNIT=3800,ADDRESS=(518,1)
 IODEVICE CUNUMBR=07F,UNIT=3800,ADDRESS=(618,1)
*
 IODEVICE CUNUMBR=080,UNIT=3278,ADDRESS=(020,8)
 IODEVICE CUNUMBR=081,UNIT=3278,ADDRESS=(120,8)
 IODEVICE CUNUMBR=082,UNIT=3278,ADDRESS=(220,8)
 IODEVICE CUNUMBR=083,UNIT=3278,ADDRESS=(320,8)
 IODEVICE CUNUMBR=084,UNIT=3278,ADDRESS=(420,8)
*
 IODEVICE CUNUMBR=085,UNIT=1403,ADDRESS=(00E,1)
 IODEVICE CUNUMBR=086,UNIT=1403,ADDRESS=(10E,1)
 IODEVICE CUNUMBR=087,UNIT=1403,ADDRESS=(20E,1)
 IODEVICE CUNUMBR=088,UNIT=1403,ADDRESS=(30E,1)
 IODEVICE CUNUMBR=089,UNIT=1403,ADDRESS=(40E,1)
 IODEVICE CUNUMBR=08A,UNIT=1403,ADDRESS=(50E,1)
 IODEVICE CUNUMBR=08B,UNIT=1403,ADDRESS=(60E,1)
*
 IODEVICE CUNUMBR=085,UNIT=1403,ADDRESS=(00F,1)
 IODEVICE CUNUMBR=086,UNIT=1403,ADDRESS=(10F,1)
 IODEVICE CUNUMBR=087,UNIT=1403,ADDRESS=(20F,1)
 IODEVICE CUNUMBR=088,UNIT=1403,ADDRESS=(30F,1)
 IODEVICE CUNUMBR=089,UNIT=1403,ADDRESS=(40F,1)
 IODEVICE CUNUMBR=08A,UNIT=1403,ADDRESS=(50F,1)
 IODEVICE CUNUMBR=08B,UNIT=1403,ADDRESS=(60F,1)
*
 IODEVICE CUNUMBR=085,UNIT=2540R,ADDRESS=(00C,1)
 IODEVICE CUNUMBR=086,UNIT=2540R,ADDRESS=(10C,1)
 IODEVICE CUNUMBR=087,UNIT=2540R,ADDRESS=(20C,1)
 IODEVICE CUNUMBR=088,UNIT=2540R,ADDRESS=(30C,1)
 IODEVICE CUNUMBR=089,UNIT=2540R,ADDRESS=(40C,1)
 IODEVICE CUNUMBR=08A,UNIT=2540R,ADDRESS=(50C,1)
 IODEVICE CUNUMBR=08B,UNIT=2540R,ADDRESS=(60C,1)
*
 IODEVICE CUNUMBR=085,UNIT=2540P,ADDRESS=(00D,1)
 IODEVICE CUNUMBR=086,UNIT=2540P,ADDRESS=(10D,1)
```

```
 IODEVICE CUNUMBR=087,UNIT=2540P,ADDRESS=(20D,1)
 IODEVICE CUNUMBR=088,UNIT=2540P,ADDRESS=(30D,1)
 IODEVICE CUNUMBR=089,UNIT=2540P,ADDRESS=(40D,1)
 IODEVICE CUNUMBR=08A,UNIT=2540P,ADDRESS=(50D,1)
 IODEVICE CUNUMBR=08B,UNIT=2540P,ADDRESS=(60D,1)
*
 IODEVICE CUNUMBR=08C,UNIT=3211,ADDRESS=(002,1)
 IODEVICE CUNUMBR=08D,UNIT=3211,ADDRESS=(102,1)
 IODEVICE CUNUMBR=08E,UNIT=3211,ADDRESS=(202,1)
 IODEVICE CUNUMBR=08F,UNIT=3211,ADDRESS=(302,1)
 IODEVICE CUNUMBR=090,UNIT=3211,ADDRESS=(402,1)
*
 IODEVICE CUNUMBR=091,UNIT=3278,ADDRESS=(040,8)
 IODEVICE CUNUMBR=092,UNIT=3278,ADDRESS=(140,8)
 IODEVICE CUNUMBR=093,UNIT=3278,ADDRESS=(240,8)
 IODEVICE CUNUMBR=094,UNIT=3278,ADDRESS=(340,8)
 IODEVICE CUNUMBR=095,UNIT=3278,ADDRESS=(440,8)
 IODEVICE CUNUMBR=096,UNIT=3278,ADDRESS=(540,8)
 IODEVICE CUNUMBR=097,UNIT=3278,ADDRESS=(640,8)
*
 IODEVICE CUNUMBR=098,UNIT=3505,ADDRESS=(012,1)
 IODEVICE CUNUMBR=099,UNIT=3505,ADDRESS=(112,1)
 IODEVICE CUNUMBR=09A,UNIT=3505,ADDRESS=(212,1)
 IODEVICE CUNUMBR=09B,UNIT=3505,ADDRESS=(312,1)
 IODEVICE CUNUMBR=09C,UNIT=3505,ADDRESS=(412,1)
*
 IODEVICE CUNUMBR=098,UNIT=3525,ADDRESS=(013,1)
 IODEVICE CUNUMBR=099,UNIT=3525,ADDRESS=(113,1)
 IODEVICE CUNUMBR=09A,UNIT=3525,ADDRESS=(213,1)
 IODEVICE CUNUMBR=09B,UNIT=3525,ADDRESS=(313,1)
 IODEVICE CUNUMBR=09C,UNIT=3525,ADDRESS=(413,1)
*
 IODEVICE CUNUMBR=09D,UNIT=3375,ADDRESS=(068,8)
 IODEVICE CUNUMBR=09E,UNIT=3375,ADDRESS=(168,8)
 IODEVICE CUNUMBR=09F,UNIT=3375,ADDRESS=(268,8)
 IODEVICE CUNUMBR=0A0,UNIT=3375,ADDRESS=(368,8)
 IODEVICE CUNUMBR=0A1,UNIT=3375,ADDRESS=(468,8)
 IODEVICE CUNUMBR=0A2,UNIT=3375,ADDRESS=(568,8)
 IODEVICE CUNUMBR=0A3,UNIT=3375,ADDRESS=(668,8)
*
 IODEVICE CUNUMBR=0A4,UNIT=3375,ADDRESS=(090,8)
 IODEVICE CUNUMBR=0A5,UNIT=3375,ADDRESS=(190,8)
 IODEVICE CUNUMBR=0A6,UNIT=3375,ADDRESS=(290,8)
 IODEVICE CUNUMBR=0A7,UNIT=3375,ADDRESS=(390,8)
 IODEVICE CUNUMBR=0A8,UNIT=3375,ADDRESS=(490,8)
*
 IODEVICE CUNUMBR=0A9,UNIT=3380,ADDRESS=(038,8)
 IODEVICE CUNUMBR=0AA,UNIT=3380,ADDRESS=(138,8)
 IODEVICE CUNUMBR=0AB,UNIT=3380,ADDRESS=(238,8)
 IODEVICE CUNUMBR=0AC,UNIT=3380,ADDRESS=(338,8)
 IODEVICE CUNUMBR=0AD,UNIT=3380,ADDRESS=(438,8)
 IODEVICE CUNUMBR=0AE,UNIT=3380,ADDRESS=(538,8)
 IODEVICE CUNUMBR=0AF,UNIT=3380,ADDRESS=(638,8)
*
 IODEVICE CUNUMBR=0B0,UNIT=3380,ADDRESS=(0A0,8)
 IODEVICE CUNUMBR=0B1,UNIT=3380,ADDRESS=(1A0,8)
 IODEVICE CUNUMBR=0B2,UNIT=3380,ADDRESS=(2A0,8)
 IODEVICE CUNUMBR=0B3,UNIT=3380,ADDRESS=(3A0,8)
 IODEVICE CUNUMBR=0B4,UNIT=3380,ADDRESS=(4A0,8)
*
```

```
 IODEVICE CUNUMBR=0B5,UNIT=3203,ADDRESS=(003,1)
 IODEVICE CUNUMBR=0B6,UNIT=3203,ADDRESS=(103,1)
 IODEVICE CUNUMBR=0B7,UNIT=3203,ADDRESS=(203,1)
 IODEVICE CUNUMBR=0B8,UNIT=3203,ADDRESS=(303,1)
*
 IODEVICE CUNUMBR=0B9,UNIT=3480,ADDRESS=(004,4)
 IODEVICE CUNUMBR=0BA,UNIT=3480,ADDRESS=(104,4)
 IODEVICE CUNUMBR=0BB,UNIT=3480,ADDRESS=(204,4)
 IODEVICE CUNUMBR=0BC,UNIT=3480,ADDRESS=(304,4)
*
 IODEVICE CUNUMBR=0BD,UNIT=3890,ADDRESS=(008,4)
 IODEVICE CUNUMBR=0BE,UNIT=3890,ADDRESS=(108,4)
*
*  5080 UNIT
*
 IODEVICE CUNUMBR=0BF,UNIT=HFGD,ADDRESS=(014,4)
 IODEVICE CUNUMBR=0C0,UNIT=HFGD,ADDRESS=(114,4)
*
```

Sample Files for 3350 Starter System

IBM provides the following sample files with the VM/XA SP 3350 starter system:

- HCPSYS ASSEMBLE for starter system
- System directory
- System residence DASD allocation
- Minidisk maps of the system residence device.

Sample HCPSYS ASSEMBLE for 3350

This is only a sample file. You need to tailor it to your installation before it can be used.

```
SYS TITLE 'HCPSYS50 - HCPSYS FOR VM/XA INSTALL BUILD'
************************************************************
***     Virtual Machine / System Product       5664-308    ***
***     Contains restricted materials of IBM               ***
***     Copyright (c) I B M   Corporation          1988    ***
***     Licensed Materials -  Property of I B M            ***
***     Refer to Copyright Instructions: Form G120-2083    ***
************************************************************
************************************************************
*
* MODULE NAME - HCPSYS
*
* DESCRIPTIVE NAME - DEFINITION OF SYSTEM
*
*
* FUNCTION - TO DEFINE THE SYSTEM
*
*
*   REGISTER CONVENTIONS - SYMBOLIC REFERENCES TO REGISTERS ARE
*                          OF THE FORM "RX" WHERE X IS A NUMBER
*                          RANGING FROM 0 TO 15 (SEE HCPEQUAT COPY)
*
*   PATCH LABEL - NONE
*
* MODULE TYPE - DATA
*
*   PROCESSOR - ASSEMBLER H (VERSION 2)
*
*   ATTRIBUTES - RESIDENT, DATA-ONLY
*
* MACROS -
*
*          SYSSTORE - DEFINE SYSTEM REAL STORAGE
*          SYSRES   - DEFINE SYSTEM RESIDENCE VOLUME
*          SYSCPVOL - DEFINE SYSTEM VOLUMES
*          SYSOPR   - DEFINE SYSTEM OPERATOR
*          SYSTIME  - DEFINE THE TIME ZONE
*          SYSACNT  - DEFINE ACCOUNTING USERID
*          SYSDUMP  - DEFINE DUMP RECEIVER USERID
*          SYSEREP  - DEFINE EREP RECEIVER USERID
*          SYSID    - DEFINE SYSTEM ID'S BY PROCESSOR
*          SYSUVOL  - DEFINE USER VOLUMES TO BE MOUNTED
*          SYSFCN   - DEFINE PRIV CLASSES ASSIGNED TO CP FUNCTIONS
*          SYSEND   - END SYSTEM GENERATION
*
```

```
*
* GENERAL COMMENTS - THIS MODULE IS THE USER'S RESPONSIBILITY.
*
*********************************************************************
         SPACE 1
*-------------------------------------------------------------
* THE SYSTEM DEFINITION MACROS SYSSTORE, SYSRES, SYSCPVOL,
* SYSOPR, AND SYSTIME, ARE CODED HERE, IN ANY ORDER
*-------------------------------------------------------------
         SPACE 1
         SYSSTORE RMSIZE=32M,VRSIZE=8M,VRFREE=256
         SYSRES SYSVOL=XASRES,                                  *
               SYSRES=123,SYSTYPE=3350,                         *
               SYSNUC=003,                                      *
               SYSCKP=(013,2),                                  *
               SYSWRM=(015,2)
         SYSTIME ZONE=5,LOC=WEST,ID=EST
         SYSOPR SYSOPER=OPERATOR
         SYSCPVOL     XASRES,XASERV,XAP001,XAP002,              *
               DRUM01,DRUM02,DRUM03,DRUM04,DRUM05,DRUM06,DRUM07,*
               PAG001,PAG002,PAG003,PAG004,PAG005,PAG006,PAG007,*
               SPL001,SPL002,SPL003,SPL004,SPL005,SPL006,SPL007
         SPACE 1
*-------------------------------------------------------------
* ANY OF THE OPTIONAL MACROS THAT ARE DESIRED ARE CODED HERE,
* IN ANY ORDER
*-------------------------------------------------------------
         SPACE 1
         SYSID      DEFAULT=INSTTST,(3081A,3081,10000)
         SYSACNT USERID=DISKACNT
         SYSDUMP USERID=OPERATNS
         SYSEREP USERID=EREP
         SYSUVOL      USRP01,USRP02,USRP03,USRP04,USRP05,USRP06,USRP07,*
               USRP08,USRP09
         SYSFCN  OPER=A,CPRD=CE,CPWT=C,SERV=F,DFLT=G
         SPACE 1
*-------------------------------------------------------------
* THE SYSEND MACRO IS CODED HERE
*-------------------------------------------------------------
         SPACE 1
         SYSEND
```

Sample Directory for 3350

This is only a sample file. You need to tailor it to your installation before it can be used.

```
*************************************************************
***     Virtual Machine / System Product       5664-308    ***
***     Contains restricted materials of IBM               ***
***     Copyright (c) I B M   Corporation          1988    ***
***     Licensed Materials -  Property of I B M            ***
***     Refer to Copyright Instructions: Form G120-2083    ***
*************************************************************
*************************************************************
*  3350 SYSTEM   VM/XA SYSTEM  PRODUCT  DIRECTORY           *
*  THE ADDRESSES 123, 124, AND 125 ARE VIRTUAL ADDRESSES.   *
*  THE ADDRESS 123 IS CRITICAL SINCE IT IS USED IN HCPSYS AND *
*  THE DIRECTORY. IF YOU WANT TO CHANGE IT, REMEMBER IT MUST  *
*  BE CHANGED IN HCPSYS, THE 'DIRECTORY' STATEMENT BELOW,     *
*  AND IN THE 'MDISK' STATEMENTS FOR THE USERID  'MAINT'.     *
*  NOTE: REMEMBER THESE ARE ONLY VIRTUAL ADDRESSES NOT REAL   *
*  ADDRESSES, SO THERE IS NO NEED TO CHANGE THEM TO MATCH     *
*  YOUR HARDWARE ADDRESSES.                                   *
*                                                             *
*  THIS DIRECTORY MUST BE MODIFIED IF MDISK PASSWORDS ARE TO  *
*  BE SPECIFIED.  WITHOUT MDISK PASSWORDS, MDISKS CANNOT BE   *
*  SHARED USING THE 'CP LINK' COMMAND.  A SAMPLE USER ENTRY   *
*  FOLLOWS: (NOTE ONLY THE MDISK STATEMENT)                   *
*                                                             *
*      USER ANYONE SOMEPASS 3M 8M FG                          *
*       AUTOLOG AUTOLOG1 OP1 MAINT                            *
*        ACCOUNT ??????                                       *
*        IPL 190                                              *
*        CONSOLE 01F 3215                                     *
*        SPOOL 00C 2540 READER A                              *
*        SPOOL 00D 2540 PUNCH A                               *
*        SPOOL 00E 1403 A                                     *
*====>   MDISK 999 33XX XXX YYY ZZZZZZ MR RPASS WPASS MPASS   *
*                                                             *
*    WHERE:  RPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                READ ONLY MODE.                              *
*            WPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                WRITE MODE.                                  *
*            MPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                MULTI-WRITE MODE.                            *
*                                                             *
*   NOTE:  EACH PASSWORD MAY BE 1-8 CHARACTERS IN LENGTH.     *
*                                                             *
*************************************************************
*                                                             *
*  Following is a sample user entry and a brief description   *
*  of each entry.  A more detailed description may be found   *
*  by referencing the VM/XA System Product Planning and       *
*  Administration manual.                                     *
*                                                             *
*    USER CMS1  abc123  3M  32M G                             *
*       -  Userid is CMS1                                     *
*       -  Logon password is abc123                           *
*       -  At logon time, virtual storage will equal 3M       *
*       -  Maximum storage that may be defined equals 32M     *
*       -  Privilege class is G                               *
*                                                             *
```

```
*     AUTOLOG AUTOLOG1 OP1 MAINT                                *
*       - Allows user CMS1 to be autolog'd by AUTOLOG1,         *
*         OP1, and MAINT                                        *
*                                                               *
*     ACCOUNT ACT4 CMSTST                                       *
*       - Time used by CMS1 will be charged to account ACT4     *
*       - Printed output will be sent to distribution CMSTST    *
*                                                               *
*     IPL CMS                                                   *
*       - Will cause automatic IPL of named save system of      *
*         CMS                                                   *
*                                                               *
*     CONSOLE 009 3215                                          *
*       - Defines virtual machines console as a 3215 at a       *
*         virtual address of 009                                *
*                                                               *
*     SPOOL 00C 2540 READER A                                   *
*     SPOOL 00D 2540 PUNCH A                                    *
*     SPOOL 00E 1403 A                                          *
*       - Defines virtual unit record devices of reader,        *
*         punch, and printer with a spooling class of 'A'       *
*                                                               *
*     LINK MAINT 190 190 RR                                     *
*       - Allows read access to minidisk 190 of user 'MAINT'    *
*                                                               *
*     LINK MAINT 19E 19E RR                                     *
*       - Allows read access to minidisk 19E of user 'MAINT'    *
*                                                               *
*     LINK MAINT 19D 19D RR                                     *
*       - Allows read access to minidisk 19d of user 'MAINT'    *
*                                                               *
*     MDISK 999 3350 236 022 CPPACK WR RPASS WPASS              *
*       - Defines minidisk with a virtual address of 999        *
*       - Specifies that the minidisk resides on a real 3350    *
*       - Minidisk starts at cylinder 236                       *
*       - Mdisk is 22 cylinders in size (starting at cyl 236    *
*       - The real volume serial number is 'CPPACK'             *
*       - Mdisk is to be accessed in write mode if no other     *
*         user has write access. Alternate access read-only     *
*       - RPASS is the required password for another user to    *
*         link to this minidisk in read mode                    *
*       - WPASS is the required password for another user to    *
*         link to this minidisk in write mode                   *
*                                                               *
*****************************************************************
*
*
*
DIRECTORY 123 3350 XASRES
*
*****************************************************************
*         SYSTEM RESERVED AREAS NOT FOR MINIDISKS               *
*****************************************************************
*
USER $ALLOC$  NOLOG
 MDISK A01 3350 000 001 XASRES  R
 MDISK B01 3350 000 001 XASERV  R
 MDISK C01 3350 000 001 XAP001  R
```

```
 MDISK D01 3350 000 001 XAP002  R
*
USER $DIRECT$ NOLOG
 MDISK A04 3350 001 002 XASRES  R
*
USER $CP-NUC$ NOLOG
 MDISK A09 3350 003 010 XASRES  R
*
USER $SYSCKP$ NOLOG
 MDISK A06 3350 013 002 XASRES  R
*
USER $SYSWRM$ NOLOG
 MDISK A07 3350 015 002 XASRES  R
*
USER $PAGE$   NOLOG
 MDISK A03 3350 017 025 XASRES R
*
USER $SPOOL$  NOLOG
 MDISK B01 3350 042 060 XASRES R
*
USER $T-DISK$ NOLOG
 MDISK B01 3350 320 020 XASRES R
*
*****************************************************************
*             CMS USERIDS                                       *
*****************************************************************
*
USER CMS1  NOLOG  3M   32M G
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT4 CMSTST
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
*********************************************************************
*                  PROP LOGICAL OPERATOR                            *
*********************************************************************
*
USER LGLOPR  NOLOG 512K 16M ABCDEG
 ACCOUNT ACT1 PROP
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3350 273 001 XASRES WR
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK OPERATOR 191 291 RR
*
*****************************************************************
*             SYSTEM RELATED USERIDS                            *
*****************************************************************
*
USER MAINT NOLOG 16M 32M ABCDEFG
```

```
AUTOLOG AUTOLOG1 OP1 MAINT
ACCOUNT 1 SYSPROG
IPL 190
NAMESAVE GCS
NAMESAVE VTAM
NAMESAVE HELP
NAMESAVE INSTHELP
CONSOLE 009 3215 T
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH  A
SPOOL 00E 1403 A
MDISK 123 3350 000 555 XASRES  MW
MDISK 124 3350 000 555 XAP001  RR
MDISK 125 3350 000 555 XASERV  RR
MDISK 126 3350 000 555 XAP002  RR

MDISK 191 3350 370 016 XASRES  MW
MDISK 193 3350 414 060 XASRES  MW
MDISK 194 3350 001 065 XASERV  MW
MDISK 201 3350 066 028 XASERV  MW RMAINT   WMAINT   MMAINT

MDISK 192 3350 094 019 XASERV  MW READ WRITE MULTIPLE
MDISK 392 3350 113 019 XASERV  MW READ WRITE MULTIPLE
MDISK 492 3350 132 013 XASERV  MW READ WRITE MULTIPLE
MDISK 692 3350 145 013 XASERV  MW READ WRITE MULTIPLE
MDISK 593 3350 158 050 XASERV  MW READ WRITE MULTIPLE
MDISK 594 3350 396 075 XAP002  MW READ WRITE MULTIPLE

MDISK 295 3350 474 038 XASRES  MW READ WRITE MULTIPLE
MDISK 395 3350 512 025 XASRES  MW READ WRITE MULTIPLE
MDISK 495 3350 280 007 XASERV  MW READ WRITE MULTIPLE
MDISK 592 3350 287 013 XASERV  MW READ WRITE MULTIPLE
MDISK 892 3350 300 007 XASERV  MW READ WRITE MULTIPLE
MDISK 491 3350 307 013 XASERV  MW READ WRITE MULTIPLE
MDISK 791 3350 320 013 XASERV  MW READ WRITE MULTIPLE
MDISK 89E 3350 333 013 XASERV  MW READ WRITE MULTIPLE
MDISK 896 3350 346 013 XASERV  MW READ WRITE MULTIPLE
MDISK 895 3350 359 013 XASERV  MW READ WRITE MULTIPLE

MDISK 293 3350 372 075 XASERV  MW RCMSAUX  WCMSAUX  MCMSAUX
MDISK 294 3350 447 100 XASERV  MW RCPAUX   WCPAUX   MCPAUX
MDISK 393 3350 326 070 XAP002  WR
MDISK 394 3350 001 163 XAP001  WR
MDISK 19C 3350 229 037 XAP001  WR
MDISK 49C 3350 266 037 XAP001  WR

MDISK 190 3350 123 074 XASRES  MW ALL
MDISK 19E 3350 303 055 XAP001  MW ALL
MDISK 19D 3350 358 037 XAP001  MW ALL

MDISK 291 3350 395 075 XAP001  MW
MDISK 391 3350 470 050 XAP001  MW
MDISK 591 3350 001 075 XAP002  MW READ WRITE MULTIPLE
MDISK 691 3350 076 050 XAP002  MW READ WRITE MULTIPLE
MDISK 49D 3350 126 037 XAP002  MW READ WRITE MULTIPLE
MDISK 501 3350 163 003 XAP002  MW READ WRITE MULTIPLE
MDISK 595 3350 166 016 XAP002  MW READ WRITE MULTIPLE
*
MDISK 490 3350 182 074 XAP002  MW
MDISK 423 3350 272 013 XAP002  MW
```

```
 MDISK 596 3350 285 025 XAP002  MW
 MDISK 59E 3350 310 013 XAP002  MW
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
 MDISK 36E 3350 323 003 XAP002 RR RPVM     WPVM     MPVM
 LINK RSCS  191 499  MW
*
USER CMSBATCH NOLOG 1M 2M G
 ACCOUNT 3 SYSTEM
 OPTION ACCT
 IPL CMS PARM AUTOCR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 195 3350 274 002 XASRES MR RBATCH   WBATCH   MBATCH
*
USER GCS NOLOG 16M 16M ABCDEFG
  AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT GCS RECVM
  IPL GCS
  NAMESAVE GCS
  CONSOLE 009 3215
  SPOOL 00C 2540 READER *
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 19D 19D RR
  LINK MAINT 595 595
  LINK MAINT 59E 59E
  MDISK 191 3350 243 004 XASRES MR RGCS WGCS MGCS
*
USER SYSMAINT   NOLOG  3M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
 IPL CMS
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 123 123  MW
 LINK MAINT 191 192  RR
 LINK MAINT 190 190  RR
 LINK MAINT 19E 19E  RR
 LINK MAINT 19D 19D  RR
*
USER OPERATOR   NOLOG 16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 2 OPERATOR
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3350 386 005 XASRES MR
 LINK OP1   191 192 RR
*
```

```
*OP1 IS AN ALTERNATE OPERATOR USERID
*
USER OP1 NOLOG 3M  32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 3 OPERATOR
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3350 391 001 XASRES MR
 LINK OPERATOR 191 192 RR
*
* This userid is for the IBM CE's use in running
* OLTSEP.
* OLTSEP is automatically IPLed in the virtual machine.
* A minimum machine size of one megabyte is required to
* run OLTSEP.
* The privilege class of F allows the CE to specify
* intensive recording mode.
* The console address of 01F is required by OLTSEP.
* The unit record addresses are those required by
* OLTSEP.
* The 5FF minidisk is the CE's OLTSEP pack.
*
USER OLTSEP NOLOG 3M 8M FG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT OLTSEP IBMCE
  IPL 5FF
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  MDISK 5FF 3350 000 555 CEPACK MR
*
*        **  EREP **
*
* This userid is for the IBM CE's use in running
* CPEREP.
* CMS is automatically IPLed in the virtual machine.
* The 190 minidisk is the CMS system disk.
* The 191 minidisk may be used to save often-used EREP
* control statements and procedures (EXECs).
* THE 190 MINIDISK HAS THE CMS TXTLIBS NECESSARY TO RUN
* EREP.
*
USER EREP NOLOG 3M 8M BFG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT EREP IBMCE
   IPL CMS
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH B
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 201 201 RR
```

```
  MDISK 191 3350 412 002 XASRES WR READ     WRITE     MULTIPLE
*
USER OPERATNS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 OPERATNS
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER D
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3350 392 017 XASRES MR RDVF     WDVF     MDVF
*
*
USER IPCS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 8 CE-ROOM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK OPERATNS 193 192 RR
*
USER AUTOLOG1   NOLOG  3M 8M ABCDEG
 AUTOLOG OP1 MAINT
 ACCOUNT 9 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 191 3350 409 001 XASRES MR RAUTOLOG WAUTOLOG MAUTOLOG
*
USER DISKACNT NOLOG 3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3350 410 001 XASRES MR READ WRITE MULTIPLE
*
USER IBMPSR NOLOG  3M 8M BG
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
```

```
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3350 411 001 XASRES MR
*
USER IVPM2 NOLOG  3M   4M G
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER VMUTIL NOLOG  3M 8M BDEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 11 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER SYSDUMP1 NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 12 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 123 3350 000 555 XASRES RR
 MDISK 124 3350 000 555 XAP001 RR
 MDISK 125 3350 000 555 XASERV RR
 MDISK 126 3350 000 555 XAP002 RR
*
*
USER RSCS    NOLOG    3M 8M BEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 13 SYSTEM
 NOPDATA
  IPL 191
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
```

```
 MDISK 191 3350 316 002 XASRES MR RRSCS    WRSCS    MRSCS
*
USER PVM     NOLOG    3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 14 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK MAINT 36E 191 MR
*
*
*****************************************************************
*             OTHER OPERATING SYSTEM USERIDS                    *
*****************************************************************
*
```

Sample XAMAINT Directory Entry for 3350

If you are using the procedure in Chapter 4, "Installing VM/XA System Product Release 2 Using an Existing VM/SP or VM/SP HPO System" on page 219 to install your system, add the following entry to your directory:

```
*
USER XAMAINT NOLOG 16M 16M ABCDEFG
 OPTION ECMODE DIAG98
 ACCOUNT 1 SYSPROG
 IPL 190
*NAMESAVE GCS
*NAMESAVE VTAM
*NAMESAVE HELP
*NAMESAVE INSTHELP
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 MDISK 123 3350 000 555 XASRES  MW
 MDISK 124 3350 000 555 XAP001  RR
 MDISK 125 3350 000 555 XASERV  RR
 MDISK 126 3350 000 555 XAP002  RR

 MDISK 191 3350 370 016 XASRES  MW
 MDISK 193 3350 414 060 XASRES  MW
 MDISK 194 3350 001 065 XASERV  MW
 MDISK 201 3350 066 028 XASERV  MW RMAINT   WMAINT   MMAINT

 MDISK 192 3350 094 019 XASERV  MW READ WRITE MULTIPLE
 MDISK 392 3350 113 019 XASERV  MW READ WRITE MULTIPLE
 MDISK 492 3350 132 013 XASERV  MW READ WRITE MULTIPLE
 MDISK 692 3350 145 013 XASERV  MW READ WRITE MULTIPLE
 MDISK 593 3350 158 050 XASERV  MW READ WRITE MULTIPLE
 MDISK 594 3350 396 075 XAP002  MW READ WRITE MULTIPLE

 MDISK 295 3350 474 038 XASRES  MW READ WRITE MULTIPLE
 MDISK 395 3350 512 025 XASRES  MW READ WRITE MULTIPLE
 MDISK 495 3350 280 007 XASERV  MW READ WRITE MULTIPLE
 MDISK 592 3350 287 013 XASERV  MW READ WRITE MULTIPLE
 MDISK 892 3350 300 007 XASERV  MW READ WRITE MULTIPLE
 MDISK 491 3350 307 013 XASERV  MW READ WRITE MULTIPLE
 MDISK 791 3350 320 013 XASERV  MW READ WRITE MULTIPLE
 MDISK 89E 3350 333 013 XASERV  MW READ WRITE MULTIPLE
 MDISK 896 3350 346 013 XASERV  MW READ WRITE MULTIPLE
 MDISK 895 3350 359 013 XASERV  MW READ WRITE MULTIPLE

 MDISK 293 3350 372 075 XASERV  MW RCMSAUX  WCMSAUX  MCMSAUX
 MDISK 294 3350 447 100 XASERV  MW RCPAUX   WCPAUX   MCPAUX
 MDISK 393 3350 326 070 XAP002  WR
 MDISK 394 3350 001 163 XAP001  WR
 MDISK 19C 3350 229 037 XAP001  WR
 MDISK 49C 3350 266 037 XAP001  WR

 MDISK 190 3350 123 074 XASRES  MW ALL
 MDISK 19E 3350 303 055 XAP001  MW ALL
 MDISK 19D 3350 358 037 XAP001  MW ALL

 MDISK 291 3350 395 075 XAP001  MW
 MDISK 391 3350 470 050 XAP001  MW
```

```
 MDISK 591 3350 001 075 XAP002  MW READ WRITE MULTIPLE
 MDISK 691 3350 076 050 XAP002  MW READ WRITE MULTIPLE
 MDISK 49D 3350 126 037 XAP002  MW READ WRITE MULTIPLE
 MDISK 501 3350 163 003 XAP002  MW READ WRITE MULTIPLE
 MDISK 595 3350 166 016 XAP002  MW READ WRITE MULTIPLE
*
 MDISK 490 3350 182 074 XAP002  MW
 MDISK 423 3350 272 013 XAP002  MW
 MDISK 596 3350 285 025 XAP002  MW
 MDISK 59E 3350 310 013 XAP002  MW
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
 MDISK 36E 3350 323 003 XAP002 RR RPVM     WPVM     MPVM
 LINK RSCS  191 499  MW
 LINK MAINT 190 390  RR
*
```

System Residence DASD Allocation for 3350

Figure 54 shows the allocation for the system residence device for the 3350 VM/XA SP Starter System.

```
TYPE CYL CYL
PERM 000-000
DRCT 001-002
PERM 003-013
PAGE 014-038
SPOL 039-077
PERM 078-205
TDSK 206-225
PERM 226-554
```

Figure 54. System Residence DASD Allocation for 3380

Minidisk Maps for 3350 System Residence Device

Figure 55 shows the minidisk maps for the system residence device for the 3350. To map your own system residence device for comparison, issue DISKMAP *fn* DIRECT, where *fn* is the name of your directory.

```
------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
CEPACK   OLTSEP      5FF     3350     0000     0554     0555

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XAP001   $ALLOC$     C01     3350     0000     0000     0001
         MAINT       394     3350     0001     0163     0163
                                       164      228       65    GAP
         MAINT       19C     3350     0229     0265     0037
         MAINT       49C     3350     0266     0302     0037
         MAINT       19E     3350     0303     0357     0055
         MAINT       19D     3350     0358     0394     0037
         MAINT       291     3350     0395     0469     0075
         MAINT       391     3350     0470     0519     0050
         MAINT       124     3350     0000     0554     0555 **OVERLAP**
         SYSDUMP1    124     3350     0000     0554     0555 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XAP002   $ALLOC$     D01     3350     0000     0000     0001
         MAINT       591     3350     0001     0075     0075
         MAINT       691     3350     0076     0125     0050
         MAINT       49D     3350     0126     0162     0037
         MAINT       501     3350     0163     0165     0003
         MAINT       595     3350     0166     0181     0016
         MAINT       490     3350     0182     0255     0074
                                       256      271       16    GAP
         MAINT       423     3350     0272     0284     0013
         MAINT       596     3350     0285     0309     0025
         MAINT       59E     3350     0310     0322     0013
         MAINT       36E     3350     0323     0325     0003
         MAINT       393     3350     0326     0395     0070
         MAINT       594     3350     0396     0470     0075
         MAINT       126     3350     0000     0554     0555 **OVERLAP**
         SYSDUMP1    126     3350     0000     0554     0555 **OVERLAP**
```

Figure 55 (Part 1 of 3). Minidisk Maps of the System Residence Device for 3350

```
------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END      SIZE
XASERV   $ALLOC$     B01     3350     0000      0000      0001
         MAINT       194     3350     0001      0065      0065
         MAINT       201     3350     0066      0093      0028
         MAINT       192     3350     0094      0112      0019
         MAINT       392     3350     0113      0131      0019
         MAINT       492     3350     0132      0144      0013
         MAINT       692     3350     0145      0157      0013
         MAINT       593     3350     0158      0207      0050
                                       208       279        72    GAP
         MAINT       495     3350     0280      0286      0007
         MAINT       592     3350     0287      0299      0013
         MAINT       892     3350     0300      0306      0007
         MAINT       491     3350     0307      0319      0013
         MAINT       791     3350     0320      0332      0013
         MAINT       89E     3350     0333      0345      0013
         MAINT       896     3350     0346      0358      0013
         MAINT       895     3350     0359      0371      0013
         MAINT       293     3350     0372      0446      0075
         MAINT       294     3350     0447      0546      0100
         MAINT       125     3350     0000      0554      0555 **OVERLAP**
         SYSDUMP1    125     3350     0000      0554      0555 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END      SIZE
XASRES   $ALLOC$     A01     3350     0000      0000      0001
         $DIRECT$    A04     3350     0001      0002      0002
         $CP-NUC$    A09     3350     0003      0012      0010
         $SYSCKP$    A06     3350     0013      0014      0002
         $SYSWRM$    A07     3350     0015      0016      0002
         $PAGE$      A03     3350     0017      0041      0025
         $SPOOL$     B01     3350     0042      0101      0060
                                       102       122        21    GAP
         MAINT       190     3350     0123      0196      0074
                                       197       242        46    GAP
```

Figure 55 (Part 2 of 3). Minidisk Maps of the System Residence Device for 3350

```
GCS          191     3350     0243     0246     0004
                              247      272       26    GAP
LGLOPR       191     3350     0273     0273     0001
CMSBATCH     195     3350     0274     0275     0002
                              276      315       40    GAP
RSCS         191     3350     0316     0317     0002
                              318      319        2    GAP
$T-DISK$     B01     3350     0320     0339     0020
                              340      369       30    GAP
MAINT        191     3350     0370     0385     0016
OPERATOR     191     3350     0386     0390     0005
OP1          191     3350     0391     0391     0001
OPERATNS     191     3350     0392     0408     0017
AUTOLOG1     191     3350     0409     0409     0001
DISKACNT     191     3350     0410     0410     0001
IBMPSR       191     3350     0411     0411     0001
EREP         191     3350     0412     0413     0002
MAINT        193     3350     0414     0473     0060
MAINT        295     3350     0474     0511     0038
MAINT        395     3350     0512     0536     0025
MAINT        123     3350     0000     0554     0555 **OVERLAP**
SYSDUMP1     123     3350     0000     0554     0555 **OVERLAP**
```

Figure 55 (Part 3 of 3). Minidisk Maps of the System Residence Device for 3350

Sample Files for 3375 Starter System

IBM provides the following sample files with the VM/XA System Product 3375 starter system:

- HCPSYS ASSEMBLE for starter system
- System directory
- System residence DASD allocation
- Minidisk maps of the system residence device.

Sample HCPSYS ASSEMBLE for 3375

This is only a sample file. You need to tailor it to your installation before it can be used.

```
SYS TITLE 'HCPSYS75 - HCPSYS FOR VM/XA INSTALL BUILD'
*****************************************************************
***     Virtual Machine / System Product       5664-308     ***
***     Contains restricted materials of IBM                ***
***     Copyright (c) I B M   Corporation          1988     ***
***     Licensed Materials -  Property of I B M             ***
***     Refer to Copyright Instructions: Form G120-2083     ***
*****************************************************************
*****************************************************************
*
* MODULE NAME - HCPSYS
*
* DESCRIPTIVE NAME - DEFINITION OF SYSTEM
*
*
* FUNCTION - TO DEFINE THE SYSTEM
*
*
*   REGISTER CONVENTIONS - SYMBOLIC REFERENCES TO REGISTERS ARE
*                          OF THE FORM "RX" WHERE X IS A NUMBER
*                          RANGING FROM 0 TO 15 (SEE HCPEQUAT COPY)
*
*   PATCH LABEL - NONE
*
* MODULE TYPE - DATA
*
*   PROCESSOR - ASSEMBLER H (VERSION 2)
*
*   ATTRIBUTES - RESIDENT, DATA-ONLY
*
* MACROS -
*
*          SYSSTORE - DEFINE SYSTEM REAL STORAGE
*          SYSRES   - DEFINE SYSTEM RESIDENCE VOLUME
*          SYSCPVOL - DEFINE SYSTEM VOLUMES
*          SYSOPR   - DEFINE SYSTEM OPERATOR
*          SYSTIME  - DEFINE THE TIME ZONE
*          SYSACNT  - DEFINE ACCOUNTING USERID
*          SYSDUMP  - DEFINE DUMP RECEIVER USERID
*          SYSEREP  - DEFINE EREP RECEIVER USERID
*          SYSID    - DEFINE SYSTEM ID'S BY PROCESSOR
*          SYSUVOL  - DEFINE USER VOLUMES TO BE MOUNTED
*          SYSFCN   - DEFINE PRIV CLASSES ASSIGNED TO CP FUNCTIONS
*          SYSEND   - END SYSTEM GENERATION
*
```

```
*
* GENERAL COMMENTS - THIS MODULE IS THE USER'S RESPONSIBILITY.
*
*********************************************************************
        SPACE 1
*-------------------------------------------------------------
* THE SYSTEM DEFINITION MACROS SYSSTORE, SYSRES, SYSCPVOL,
* SYSOPR, AND SYSTIME, ARE CODED HERE, IN ANY ORDER
*-------------------------------------------------------------
        SPACE 1
        SYSSTORE RMSIZE=32M,VRSIZE=8M,VRFREE=256
        SYSRES SYSVOL=XASRES,                                   *
              SYSRES=123,SYSTYPE=3375,                          *
              SYSNUC=005,                                       *
              SYSCKP=(021,4),                                   *
              SYSWRM=(025,4)
        SYSTIME ZONE=5,LOC=WEST,ID=EST
        SYSOPR SYSOPER=OPERATOR
        SYSCPVOL     XASRES,XASERV,XAP001,                      *
              DRUM01,DRUM02,DRUM03,DRUM04,DRUM05,DRUM06,DRUM07, *
              PAG001,PAG002,PAG003,PAG004,PAG005,PAG006,PAG007, *
              SPL001,SPL002,SPL003,SPL004,SPL005,SPL006,SPL007
        SPACE 1
*-------------------------------------------------------------
* ANY OF THE OPTIONAL MACROS THAT ARE DESIRED ARE CODED HERE,
* IN ANY ORDER
*-------------------------------------------------------------
        SPACE 1
        SYSID      DEFAULT=INSTTST,(3081A,3081,10000)
        SYSACNT USERID=DISKACNT
        SYSDUMP USERID=OPERATNS
        SYSEREP USERID=EREP
        SYSUVOL      USRP01,USRP02,USRP03,USRP04,USRP05,USRP06,USRP07,*
              USRP08,USRP09
        SYSFCN  OPER=A,CPRD=CE,CPWT=C,SERV=F,DFLT=G
        SPACE 1
*-------------------------------------------------------------
* THE SYSEND MACRO IS CODED HERE
*-------------------------------------------------------------
        SPACE 1
        SYSEND
```

Sample Directory for 3375

This is only a sample file. You need to tailor it to your installation before it can be used.

```
************************************************************
***     Virtual Machine / System Product       5664-308    ***
***     Contains restricted materials of IBM               ***
***     Copyright (c) I B M   Corporation          1988    ***
***     Licensed Materials -  Property of I B M            ***
***     Refer to Copyright Instructions: Form G120-2083    ***
************************************************************
************************************************************
*  3375 SYSTEM   VM/XA SYSTEM  PRODUCT  DIRECTORY            *
*  THE ADDRESSES 123, 124, AND 125 ARE VIRTUAL ADDRESSES.    *
*  THE ADDRESS 123 IS CRITICAL SINCE IT IS USED IN HCPSYS AND *
*  THE DIRECTORY. IF YOU WANT TO CHANGE IT, REMEMBER IT MUST  *
*  BE CHANGED IN HCPSYS, THE 'DIRECTORY' STATEMENT BELOW,     *
*  AND IN THE 'MDISK' STATEMENTS FOR THE USERID  'MAINT'.     *
*  NOTE: REMEMBER THESE ARE ONLY VIRTUAL ADDRESSES NOT REAL   *
*  ADDRESSES, SO THERE IS NO NEED TO CHANGE THEM TO MATCH     *
*  YOUR HARDWARE ADDRESSES.                                   *
*                                                             *
*  THIS DIRECTORY MUST BE MODIFIED IF MDISK PASSWORDS ARE TO  *
*  BE SPECIFIED.  WITHOUT MDISK PASSWORDS, MDISKS CANNOT BE   *
*  SHARED USING THE 'CP LINK' COMMAND.  A SAMPLE USER ENTRY   *
*  FOLLOWS: (NOTE ONLY THE MDISK STATEMENT)                   *
*                                                             *
*      USER ANYONE SOMEPASS 3M 8M FG                          *
*       AUTOLOG AUTOLOG1 OP1 MAINT                            *
*        ACCOUNT ??????                                       *
*        IPL 190                                              *
*        CONSOLE 01F 3215                                     *
*        SPOOL 00C 2540 READER A                              *
*        SPOOL 00D 2540 PUNCH A                               *
*        SPOOL 00E 1403 A                                     *
*====>   MDISK 999 33XX XXX YYY ZZZZZZ MR RPASS WPASS MPASS   *
*                                                             *
*    WHERE:  RPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                READ ONLY MODE.                              *
*            WPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                WRITE MODE.                                  *
*            MPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                MULTI-WRITE MODE.                            *
*                                                             *
*   NOTE:  EACH PASSWORD MAY BE 1-8 CHARACTERS IN LENGTH.     *
*                                                             *
*                                                             *
************************************************************
*                                                            *
*  Following is a sample user entry and a brief description  *
*  of each entry.  A more detailed description may be found  *
*  by referencing the VM/XA System Product Planning and      *
*  Administration manual.                                    *
*                                                            *
*    USER CMS1  abc123  3M  32M G                            *
*       -  Userid is CMS1                                    *
*       -  Logon password is abc123                          *
*       -  At logon time, virtual storage will equal 3M      *
*       -  Maximum storage that may be defined equals 32M    *
*       -  Privilege class is G                              *
```

```
*                                                                    *
*     AUTOLOG AUTOLOG1 OP1 MAINT                                     *
*       - Allows user CMS1 to be autolog'd by AUTOLOG1,              *
*         OP1, and MAINT                                             *
*                                                                    *
*     ACCOUNT ACT4 CMSTST                                            *
*       - Time used by CMS1 will be charged to account ACT4          *
*       - Printed output will be sent to distribution CMSTST         *
*                                                                    *
*     IPL CMS                                                        *
*       - Will cause automatic IPL of named save system of           *
*         CMS                                                        *
*                                                                    *
*     CONSOLE 009 3215                                               *
*       - Defines virtual machines console as a 3215 at a            *
*         virtual address of 009                                     *
*                                                                    *
*     SPOOL 00C 2540 READER A                                        *
*     SPOOL 00D 2540 PUNCH A                                         *
*     SPOOL 00E 1403 A                                               *
*       - Defines virtual unit record devices of reader,             *
*         punch, and printer with a spooling class of 'A'            *
*                                                                    *
*     LINK MAINT 190 190 RR                                          *
*       - Allows read access to minidisk 190 of user 'MAINT'         *
*                                                                    *
*     LINK MAINT 19E 19E RR                                          *
*       - Allows read access to minidisk 19E of user 'MAINT'         *
*                                                                    *
*     LINK MAINT 19D 19D RR                                          *
*       - Allows read access to minidisk 19d of user 'MAINT'         *
*                                                                    *
*     MDISK 999 3350 236 022 CPPACK WR RPASS WPASS                   *
*       - Defines minidisk with a virtual address of 999             *
*       - Specifies that the minidisk resides on a real 3350         *
*       - Minidisk starts at cylinder 236                            *
*       - Mdisk is 22 cylinders in size (starting at cyl 236         *
*       - The real volume serial number is 'CPPACK'                  *
*       - Mdisk is to be accessed in write mode if no other          *
*         user has write access. Alternate access read-only          *
*       - RPASS is the required password for another user to         *
*         link to this minidisk in read mode                         *
*       - WPASS is the required password for another user to         *
*         link to this minidisk in write mode                        *
**********************************************************************
*
*
*
DIRECTORY 123 3375 XASRES
*
**********************************************************************
*         SYSTEM RESERVED AREAS NOT FOR MINIDISKS                    *
**********************************************************************
*
USER $ALLOC$  NOLOG
 MDISK A01 3375 000 001 XASRES R
 MDISK B01 3375 000 001 XASERV R
 MDISK C01 3375 000 001 XAP001 R
*
USER $DIRECT$ NOLOG
```

```
 MDISK A04 3375 001 004 XASRES  R
*
USER $CP-NUC$ NOLOG
 MDISK A09 3375 005 016 XASRES  R
*
USER $SYSCKP$ NOLOG
 MDISK A06 3375 021 004 XASRES  R
*
USER $SYSWRM$ NOLOG
 MDISK A07 3375 025 004 XASRES  R
*
USER $PAGE$   NOLOG
 MDISK A03 3375 029 038 XASRES R
*
USER $SPOOL$  NOLOG
 MDISK B01 3375 067 092 XASRES R
*
USER $T-DISK$ NOLOG
 MDISK B01 3375 561 031 XASRES R
*
*****************************************************************
*             CMS USERIDS                                       *
*****************************************************************
*
USER CMS1  NOLOG  3M   32M G
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT4 CMSTST
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
*****************************************************************
*                  PROP LOGICAL OPERATOR                        *
*****************************************************************
*
USER LGLOPR  NOLOG 512K 16M ABCDEG
 ACCOUNT ACT1 PROP
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3375 487 001 XASRES WR
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK OPERATOR 191 291 RR
*
*****************************************************************
*             SYSTEM RELATED USERIDS                            *
*****************************************************************
*
USER MAINT NOLOG 16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
 IPL 190
```

```
 NAMESAVE GCS
 NAMESAVE VTAM
 NAMESAVE HELP
 NAMESAVE INSTHELP
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 MDISK 123 3375 000 959 XASRES  MW
 MDISK 124 3375 000 959 XAP001  RR
 MDISK 125 3375 000 959 XASERV  RR
*
 MDISK 191 3375 671 025 XASRES  MW
 MDISK 193 3375 742 076 XASRES  MW
 MDISK 194 3375 818 082 XASRES  MW
 MDISK 201 3375 067 035 XASERV  MW RMAINT   WMAINT   MMAINT
*
 MDISK 192 3375 102 024 XASERV  MW READ WRITE MULTIPLE
 MDISK 392 3375 126 024 XASERV  MW READ WRITE MULTIPLE
 MDISK 492 3375 150 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 692 3375 166 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 594 3375 182 094 XASERV  MW READ WRITE MULTIPLE
 MDISK 593 3375 276 063 XASERV  MW READ WRITE MULTIPLE
 MDISK 423 3375 339 016 XASERV  MW
*
 MDISK 295 3375 492 047 XASRES  MW READ WRITE MULTIPLE
 MDISK 395 3375 447 032 XASRES  MW READ WRITE MULTIPLE
 MDISK 495 3375 434 008 XASERV  MW READ WRITE MULTIPLE
 MDISK 592 3375 442 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 892 3375 458 008 XASERV  MW READ WRITE MULTIPLE
 MDISK 491 3375 466 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 791 3375 482 016 XASERV  MW
 MDISK 89E 3375 498 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 896 3375 514 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 895 3375 530 016 XASERV  MW READ WRITE MULTIPLE

 MDISK 293 3375 589 094 XASERV  MW RCMSAUX  WCMSAUX  MCMSAUX
 MDISK 294 3375 683 125 XASERV  MW RCPAUX   WCPAUX   MCPAUX
 MDISK 393 3375 001 088 XAP001  WR
 MDISK 394 3375 748 204 XAP001  WR
 MDISK 19C 3375 173 046 XAP001  WR
 MDISK 49C 3375 219 046 XAP001  WR
 MDISK 596 3375 265 032 XAP001  WR
 MDISK 59E 3375 297 016 XAP001  WR
*
 MDISK 190 3375 223 113 XASRES  MW ALL
 MDISK 19E 3375 313 069 XAP001  MW ALL
 MDISK 19D 3375 382 046 XAP001  MW ALL
*
 MDISK 291 3375 336 094 XASRES  MW
 MDISK 391 3375 159 063 XASRES  MW
 MDISK 591 3375 428 094 XAP001  MW READ WRITE MULTIPLE
 MDISK 691 3375 522 063 XAP001  MW READ WRITE MULTIPLE
 MDISK 49D 3375 585 046 XAP001  MW READ WRITE MULTIPLE
 MDISK 490 3375 631 113 XAP001  MW
 MDISK 501 3375 744 004 XAP001  MW READ WRITE MULTIPLE
 MDISK 595 3375 546 020 XASERV  MW READ WRITE MULTIPLE
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
 MDISK 36E 3375 089 004 XAP001 RR RPVM     WPVM     MPVM
```

```
 LINK RSCS  191 499  MW
*
USER CMSBATCH NOLOG 1M 2M G
 ACCOUNT 3 SYSTEM
 OPTION ACCT
 IPL CMS PARM AUTOCR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 195 3375 488 004 XASRES MR RBATCH   WBATCH   MBATCH
*
USER GCS NOLOG 16M 16M ABCDEFG
  AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT GCS RECVM
  IPL GCS
  NAMESAVE GCS
  CONSOLE 009 3215
  SPOOL 00C 2540 READER *
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 19D 19D RR
  LINK MAINT 595 595
  LINK MAINT 59E 59E
  MDISK 191 3375 441 006 XASRES MR RGCS WGCS MGCS
*
USER SYSMAINT  NOLOG   3M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
 IPL CMS
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 123 123  MW
 LINK MAINT 191 192  RR
 LINK MAINT 190 190  RR
 LINK MAINT 19E 19E  RR
 LINK MAINT 19D 19D  RR
*
USER OPERATOR NOLOG   16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 2 OPERATOR
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3375 696 008 XASRES MR
 LINK OP1   191 192 RR
*
*OP1 IS AN ALTERNATE OPERATOR USERID
*
USER OP1 NOLOG 3M  32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 3 OPERATOR
```

```
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3375 704 002 XASRES MR
 LINK OPERATOR 191 192 RR
*
* This userid is for the IBM CE's use in running
* OLTSEP.
* OLTSEP is automatically IPLed in the virtual machine.
* A minimum machine size of one megabyte is required to
* run OLTSEP.
* The privilege class of F allows the CE to specify
* intensive recording mode.
* The console address of 01F is required by OLTSEP.
* The unit record addresses are those required by
* OLTSEP.
* The 5FF minidisk is the CE's OLTSEP pack.
*
USER OLTSEP NOLOG 3M 8M FG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT OLTSEP IBMCE
  IPL 5FF
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  MDISK 5FF 3375 000 959 CEPACK MR
*
*       **  EREP **
*
* This userid is for the IBM CE's use in running
* CPEREP.
* CMS is automatically IPLed in the virtual machine.
* The 190 minidisk is the CMS system disk.
* The 191 minidisk may be used to save often-used EREP
* control statements and procedures (EXECs).
* THE 190 MINIDISK HAS THE CMS TXTLIBS NECESSARY TO RUN
* EREP.
*
USER EREP NOLOG 3M 8M BFG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT EREP IBMCE
   IPL CMS
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH B
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 201 201 RR
  MDISK 191 3375 738 004 XASRES WR READ     WRITE   MULTIPLE
*
USER OPERATNS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 OPERATNS
 IPL CMS
```

```
 CONSOLE 009 3215
 SPOOL 00C 2540 READER D
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3375 706 026 XASRES MR RDVF     WDVF     MDVF
*
*
USER IPCS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 8 CE-ROOM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK OPERATNS 193 192 RR
*
USER AUTOLOG1  NOLOG   3M 8M ABCDEG
 AUTOLOG OP1 MAINT
 ACCOUNT 9 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 191 3375 732 002 XASRES MR RAUTOLOG WAUTOLOG MAUTOLOG
*
USER DISKACNT NOLOG 3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3375 734 002 XASRES MR READ WRITE MULTIPLE
*
USER IBMPSR NOLOG  3M 8M BG
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3375 736 002 XASRES WR
*
USER IVPM2 NOLOG  3M   4M G
```

```
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER VMUTIL NOLOG  3M 8M BDEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 11 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER SYSDUMP1 NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 12 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 123 3375 000 959 XASRES RR
 MDISK 124 3375 000 959 XAP001 RR
 MDISK 125 3375 000 959 XASERV RR
*
*
USER RSCS    NOLOG    3M 8M BEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 13 SYSTEM
 NOPDATA
  IPL 191
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3375 553 004 XASRES MR RRSCS    WRSCS    MRSCS
*
USER PVM      NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 14 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
```

```
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK MAINT 36E 191 MR
*
*
*****************************************************************
*             OTHER OPERATING SYSTEM USERIDS                    *
*****************************************************************
*
```

Sample XAMAINT Directory Entry for 3375

If you are using the procedure in Chapter 4, "Installing VM/XA System Product Release 2 Using an Existing VM/SP or VM/SP HPO System" on page 219 to install your system, add the following entry to your directory:

```
*
USER XAMAINT NOLOG 16M 16M ABCDEFG
 OPTION ECMODE DIAG98
 ACCOUNT 1 SYSPROG
 IPL 190
*NAMESAVE GCS
*NAMESAVE VTAM
*NAMESAVE HELP
*NAMESAVE INSTHELP
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 MDISK 123 3375 000 959 XASRES  MW
 MDISK 124 3375 000 959 XAP001  RR
 MDISK 125 3375 000 959 XASERV  RR
*
 MDISK 191 3375 671 025 XASRES  MW
 MDISK 193 3375 742 076 XASRES  MW
 MDISK 194 3375 818 082 XASRES  MW
 MDISK 201 3375 067 035 XASERV  MW RMAINT   WMAINT   MMAINT
*
 MDISK 192 3375 102 024 XASERV  MW READ WRITE MULTIPLE
 MDISK 392 3375 126 024 XASERV  MW READ WRITE MULTIPLE
 MDISK 492 3375 150 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 692 3375 166 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 594 3375 182 094 XASERV  MW READ WRITE MULTIPLE
 MDISK 593 3375 276 063 XASERV  MW READ WRITE MULTIPLE
 MDISK 423 3375 339 016 XASERV  MW
*
 MDISK 295 3375 492 047 XASRES  MW READ WRITE MULTIPLE
 MDISK 395 3375 447 032 XASRES  MW READ WRITE MULTIPLE
 MDISK 495 3375 434 008 XASERV  MW READ WRITE MULTIPLE
 MDISK 592 3375 442 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 892 3375 458 008 XASERV  MW READ WRITE MULTIPLE
 MDISK 491 3375 466 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 791 3375 482 016 XASERV  MW
 MDISK 89E 3375 498 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 896 3375 514 016 XASERV  MW READ WRITE MULTIPLE
 MDISK 895 3375 530 016 XASERV  MW READ WRITE MULTIPLE

 MDISK 293 3375 589 094 XASERV  MW RCMSAUX  WCMSAUX  MCMSAUX
 MDISK 294 3375 683 125 XASERV  MW RCPAUX   WCPAUX   MCPAUX
 MDISK 393 3375 001 088 XAP001  WR
 MDISK 394 3375 748 204 XAP001  WR
 MDISK 19C 3375 173 046 XAP001  WR
 MDISK 49C 3375 219 046 XAP001  WR
 MDISK 596 3375 265 032 XAP001  WR
 MDISK 59E 3375 297 016 XAP001  WR
*
 MDISK 190 3375 223 113 XASRES  MW ALL
 MDISK 19E 3375 313 069 XAP001  MW ALL
 MDISK 19D 3375 382 046 XAP001  MW ALL
*
```

```
 MDISK 291 3375 336 094 XASRES  MW
 MDISK 391 3375 159 063 XASRES  MW
 MDISK 591 3375 428 094 XAP001  MW READ WRITE MULTIPLE
 MDISK 691 3375 522 063 XAP001  MW READ WRITE MULTIPLE
 MDISK 49D 3375 585 046 XAP001  MW READ WRITE MULTIPLE
 MDISK 490 3375 631 113 XAP001  MW
 MDISK 501 3375 744 004 XAP001  MW READ WRITE MULTIPLE
 MDISK 595 3375 546 020 XASERV  MW READ WRITE MULTIPLE
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
 MDISK 36E 3375 089 004 XAP001 RR RPVM     WPVM     MPVM
 LINK RSCS  191 499  MW
 LINK MAINT 190 390  RR
*
```

System Residence DASD Allocation for 3375

Figure 56 shows the allocation for the system residence device for the 3375 VM/XA SP starter system.

```
TYPE CYL CYL
PERM 000-000
DRCT 001-003
PERM 004-018
PAGE 019-054
SPOL 055-113
PERM 114-306
TDSK 307-336
PERM 337-958
```

Figure 56. System Residence DASD Allocation for 3375

Minidisk Maps for 3375 System Residence Device

Figure 57 shows the minidisk maps of the system residence device for the 3375. To map your own system residence device for comparison, issue DISKMAP *fn* DIRECT, where *fn* is the name of your directory.

```
------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
CEPACK   OLTSEP      5FF     3375     0000     0958     0959

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XAP001   $ALLOC$     C01     3375     0000     0000     0001
         MAINT       393     3375     0001     0088     0088
         MAINT       36E     3375     0089     0092     0004
                                        93      172       80    GAP
         MAINT       19C     3375     0173     0218     0046
         MAINT       49C     3375     0219     0264     0046
         MAINT       596     3375     0265     0296     0032
         MAINT       59E     3375     0297     0312     0016
         MAINT       19E     3375     0313     0381     0069
         MAINT       19D     3375     0382     0427     0046
         MAINT       591     3375     0428     0521     0094
         MAINT       691     3375     0522     0584     0063
         MAINT       49D     3375     0585     0630     0046
         MAINT       490     3375     0631     0743     0113
         MAINT       501     3375     0744     0747     0004
         MAINT       394     3375     0748     0951     0204
         MAINT       124     3375     0000     0958     0959 **OVERLAP**
         SYSDUMP1    124     3375     0000     0958     0959 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XASERV   $ALLOC$     B01     3375     0000     0000     0001
                                         1       66       66    GAP
         MAINT       201     3375     0067     0101     0035
         MAINT       192     3375     0102     0125     0024
         MAINT       392     3375     0126     0149     0024
         MAINT       492     3375     0150     0165     0016
         MAINT       692     3375     0166     0181     0016
         MAINT       594     3375     0182     0275     0094
         MAINT       593     3375     0276     0338     0063
         MAINT       423     3375     0339     0354     0016
                                       355      433       79    GAP
```

Figure 57 (Part 1 of 3). Minidisk Maps of the System Residence Device for 3375

```
         MAINT      495     3375     0434     0441     0008
         MAINT      592     3375     0442     0457     0016
         MAINT      892     3375     0458     0465     0008
         MAINT      491     3375     0466     0481     0016
         MAINT      791     3375     0482     0497     0016
         MAINT      89E     3375     0498     0513     0016
         MAINT      896     3375     0514     0529     0016
         MAINT      895     3375     0530     0545     0016
         MAINT      595     3375     0546     0565     0020
                                      566      588       23    GAP
         MAINT      293     3375     0589     0682     0094
         MAINT      294     3375     0683     0807     0125
         MAINT      125     3375     0000     0958     0959 **OVERLAP**
         SYSDUMP1   125     3375     0000     0958     0959 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID     CUU   DEVTYPE   START     END      SIZE
XASRES   $ALLOC$    A01     3375     0000     0000     0001
         $DIRECT$   A04     3375     0001     0004     0004
         $CP-NUC$   A09     3375     0005     0020     0016
         $SYSCKP$   A06     3375     0021     0024     0004
         $SYSWRM$   A07     3375     0025     0028     0004
         $PAGE$     A03     3375     0029     0066     0038
         $SPOOL$    B01     3375     0067     0158     0092
         MAINT      391     3375     0159     0221     0063
                                      222      222        1    GAP
         MAINT      190     3375     0223     0335     0113
         MAINT      291     3375     0336     0429     0094
                                      430      440       11    GAP
         GCS        191     3375     0441     0446     0006
         MAINT      395     3375     0447     0478     0032
                                      479      486        8    GAP
         LGLOPR     191     3375     0487     0487     0001
         CMSBATCH   195     3375     0488     0491     0004
         MAINT      295     3375     0492     0538     0047
                                      539      552       14    GAP
         RSCS       191     3375     0553     0556     0004
                                      557      560        4    GAP
```

Figure 57 (Part 2 of 3). Minidisk Maps of the System Residence Device for 3375

```
$T-DISK$    B01     3375    0561    0591    0031
                            592     670      79    GAP
MAINT       191     3375    0671    0695    0025
OPERATOR    191     3375    0696    0703    0008
OP1         191     3375    0704    0705    0002
OPERATNS    191     3375    0706    0731    0026
AUTOLOG1    191     3375    0732    0733    0002
DISKACNT    191     3375    0734    0735    0002
IBMPSR      191     3375    0736    0737    0002
EREP        191     3375    0738    0741    0004
MAINT       193     3375    0742    0817    0076
MAINT       194     3375    0818    0899    0082
MAINT       123     3375    0000    0958    0959 **OVERLAP**
SYSDUMP1    123     3375    0000    0958    0959 **OVERLAP**
```

Figure 57 (Part 3 of 3). Minidisk Maps of the System Residence Device for 3375

Sample Files for 3380 Starter System

IBM provides the following sample files with the VM/XA SP 3380 Starter System:

- HCPSYS ASSEMBLE for Starter System
- System directories for 3380, 3380-E4, and 3380-K
- System residence DASD allocation
- Minidisk maps of the system residence device.

Sample HCPSYS ASSEMBLE for 3380

This is only a sample file. You need to tailor it to your installation before it can be used.

```
SYS TITLE 'HCPSYS80 - HCPSYS FOR VM/XA INSTALL BUILD'
*********************************************************************
* THIS MODULE IS RESTRICTED MATERIALS OF IBM.                       *
* 5664-308 (C) COPYRIGHT IBM CORPORATION - 1988                     *
* LICENSED MATERIALS - PROPERTY OF IBM                              *
* SEE COPYRIGHT INSTRUCTIONS, G120-2083                             *
*********************************************************************
*************************************************************
*
* MODULE NAME - HCPSYS
*
* DESCRIPTIVE NAME - DEFINITION OF SYSTEM
*
*
* FUNCTION - TO DEFINE THE SYSTEM
*
*
*   REGISTER CONVENTIONS - SYMBOLIC REFERENCES TO REGISTERS ARE
*                          OF THE FORM "RX" WHERE X IS A NUMBER
*                          RANGING FROM 0 TO 15 (SEE HCPEQUAT COPY)
*
*   PATCH LABEL - NONE
*
* MODULE TYPE - DATA
*
*   PROCESSOR - ASSEMBLER H (VERSION 2)
*
*   ATTRIBUTES - RESIDENT, DATA-ONLY
*
* MACROS -
*
*          SYSSTORE - DEFINE SYSTEM REAL STORAGE
*          SYSRES   - DEFINE SYSTEM RESIDENCE VOLUME
*          SYSCPVOL - DEFINE SYSTEM VOLUMES
*          SYSOPR   - DEFINE SYSTEM OPERATOR
*          SYSTIME  - DEFINE THE TIME ZONE
*          SYSACNT  - DEFINE ACCOUNTING USERID
*          SYSDUMP  - DEFINE DUMP RECEIVER USERID
*          SYSEREP  - DEFINE EREP RECEIVER USERID
*          SYSID    - DEFINE SYSTEM ID'S BY PROCESSOR
*          SYSUVOL  - DEFINE USER VOLUMES TO BE MOUNTED
*          SYSFCN   - DEFINE PRIV CLASSES ASSIGNED TO CP FUNCTIONS
*          SYSEND   - END SYSTEM GENERATION
*
*
```

```
* GENERAL COMMENTS - THIS MODULE IS THE USER'S RESPONSIBILITY.
*
************************************************************************
        SPACE 1
*--------------------------------------------------------------
* THE SYSTEM DEFINITION MACROS SYSSTORE, SYSRES, SYSCPVOL,
* SYSOPR, AND SYSTIME, ARE CODED HERE, IN ANY ORDER
*--------------------------------------------------------------
        SPACE 1
        SYSSTORE RMSIZE=32M,VRSIZE=8M,VRFREE=256
        SYSRES SYSVOL=XASRES,                                    *
              SYSRES=123,SYSTYPE=3380,                           *
              SYSNUC=003,                                        *
              SYSCKP=(013,2),                                    *
              SYSWRM=(015,2)
        SYSTIME ZONE=5,LOC=WEST,ID=EST
        SYSOPR SYSOPER=OPERATOR
        SYSCPVOL     XASRES,XASERV,XAP001,                       *
              DRUM01,DRUM02,DRUM03,DRUM04,DRUM05,DRUM06,DRUM07,  *
              PAG001,PAG002,PAG003,PAG004,PAG005,PAG006,PAG007,  *
              SPL001,SPL002,SPL003,SPL004,SPL005,SPL006,SPL007
        SPACE 1
*--------------------------------------------------------------
* ANY OF THE OPTIONAL MACROS THAT ARE DESIRED ARE CODED HERE,
* IN ANY ORDER
*--------------------------------------------------------------
        SPACE 1
        SYSID      DEFAULT=INSTTST,(3081A,3081,10000)
        SYSACNT USERID=DISKACNT
        SYSDUMP USERID=OPERATNS
        SYSEREP USERID=EREP
        SYSUVOL      USRP01,USRP02,USRP03,USRP04,USRP05,USRP06,USRP07,*
              USRP08,USRP09
        SYSFCN  OPER=A,CPRD=CE,CPWT=C,SERV=F,DFLT=G
        SPACE 1
*--------------------------------------------------------------
* THE SYSEND MACRO IS CODED HERE
*--------------------------------------------------------------
        SPACE 1
        SYSEND
```

There are three sets of sample directory, minidisk allocations, and minidisk maps for different models of 3380 DASD. The first set ("Sample Directory for 3380," "System Residence DASD Allocation for 3380" on page 814 , and "Minidisk Maps for 3380 System Residence Device" on page 815) is for the 3380. The second set ("Sample Directory for 3380-E4" on page 817, "System Residence DASD Allocation for 3380-E4" on page 828, and "Minidisk Maps for 3380-E4 System Residence Device" on page 829) is for the 3380-E4. The third set ("Sample Directory for 3380-K" on page 832, "System Residence DASD Allocation for 3380-K" on page 843, and "Minidisk Maps for 3380-K System Residence Device" on page 844) is for the 3380-K.

Sample Directory for 3380

This is only a sample file. You need to tailor it to your installation before it can be used.

```
************************************************************
***     Virtual Machine / System Product       5664-308    ***
***     Contains restricted materials of IBM               ***
***     Copyright (c) I B M   Corporation          1988    ***
***     Licensed Materials -  Property of I B M            ***
***     Refer to Copyright Instructions: Form G120-2083    ***
************************************************************
************************************************************
*  3380 SYSTEM   VM/XA SYSTEM  PRODUCT  DIRECTORY            *
*  THE ADDRESSES 123, 124, AND 125 ARE VIRTUAL ADDRESSES.    *
*  THE ADDRESS 123 IS CRITICAL SINCE IT IS USED IN HCPSYS AND *
*  THE DIRECTORY. IF YOU WANT TO CHANGE IT, REMEMBER IT MUST  *
*  BE CHANGED IN HCPSYS, THE 'DIRECTORY' STATEMENT BELOW,     *
*  AND IN THE 'MDISK' STATEMENTS FOR THE USERID  'MAINT'.     *
*  NOTE: REMEMBER THESE ARE ONLY VIRTUAL ADDRESSES NOT REAL   *
*  ADDRESSES, SO THERE IS NO NEED TO CHANGE THEM TO MATCH     *
*  YOUR HARDWARE ADDRESSES.                                   *
*                                                             *
*  THIS DIRECTORY MUST BE MODIFIED IF MDISK PASSWORDS ARE TO  *
*  BE SPECIFIED.  WITHOUT MDISK PASSWORDS, MDISKS CANNOT BE   *
*  SHARED USING THE 'CP LINK' COMMAND.  A SAMPLE USER ENTRY   *
*  FOLLOWS: (NOTE ONLY THE MDISK STATEMENT)                   *
*                                                             *
*      USER ANYONE SOMEPASS 3M 8M FG                          *
*       AUTOLOG AUTOLOG1 OP1 MAINT                            *
*        ACCOUNT ??????                                       *
*        IPL 190                                              *
*        CONSOLE 01F 3215                                     *
*        SPOOL 00C 2540 READER A                              *
*        SPOOL 00D 2540 PUNCH A                               *
*        SPOOL 00E 1403 A                                     *
*====>   MDISK 999 33XX XXX YYY ZZZZZZ MR RPASS WPASS MPASS   *
*                                                             *
*    WHERE:  RPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                READ ONLY MODE.                              *
*            WPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                WRITE MODE.                                  *
*            MPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                MULTI-WRITE MODE.                            *
*                                                             *
*   NOTE:  EACH PASSWORD MAY BE 1-8 CHARACTERS IN LENGTH.     *
*                                                             *
************************************************************
*                                                             *
*                                                             *
*  Following is a sample user entry and a brief description   *
```

```
*  of each entry.  A more detailed description may be found   *
*  by referencing the VM/XA System Product Planning and       *
*  Administration manual.                                     *
*                                                             *
*    USER CMS1  abc123  3M  32M G                             *
*       - Userid is CMS1                                      *
*       - Logon password is abc123                            *
*       - At logon time, virtual storage will equal 3M        *
*       - Maximum storage that may be defined equals 32M      *
*       - Privilege class is G                                *
*                                                             *
*     AUTOLOG AUTOLOG1 OP1 MAINT                              *
*       - Allows user CMS1 to be autolog'd by AUTOLOG1,       *
*         OP1, and MAINT                                      *
*                                                             *
*     ACCOUNT ACT4 CMSTST                                     *
*       - Time used by CMS1 will be charged to account ACT4   *
*       - Printed output will be sent to distribution CMSTST  *
*                                                             *
*     IPL CMS                                                 *
*       - Will cause automatic IPL of named save system of    *
*         CMS                                                 *
*                                                             *
*     CONSOLE 009 3215                                        *
*       - Defines virtual machines console as a 3215 at a     *
*         virtual address of 009                              *
*                                                             *
*     SPOOL 00C 2540 READER A                                 *
*     SPOOL 00D 2540 PUNCH A                                  *
*     SPOOL 00E 1403 A                                        *
*       - Defines virtual unit record devices of reader,      *
*         punch, and printer with a spooling class of 'A'     *
*                                                             *
*     LINK MAINT 190 190 RR                                   *
*       - Allows read access to minidisk 190 of user 'MAINT'  *
*                                                             *
*     LINK MAINT 19E 19E RR                                   *
*       - Allows read access to minidisk 19E of user 'MAINT'  *
*                                                             *
*     LINK MAINT 19D 19D RR                                   *
*       - Allows read access to minidisk 19d of user 'MAINT'  *
*                                                             *
*     MDISK 999 3350 236 022 CPPACK WR RPASS WPASS            *
*       - Defines minidisk with a virtual address of 999      *
*       - Specifies that the minidisk resides on a real 3350  *
*       - Minidisk starts at cylinder 236                     *
*       - Mdisk is 22 cylinders in size (starting at cyl 236  *
*       - The real volume serial number is 'CPPACK'           *
*       - Mdisk is to be accessed in write mode if no other   *
*         user has write access. Alternate access read-only   *
*       - RPASS is the required password for another user to  *
*         link to this minidisk in read mode                  *
*       - WPASS is the required password for another user to  *
*         link to this minidisk in write mode                 *
*************************************************************
*
*
*
DIRECTORY 123 3380 XASRES
*
```

```
*****************************************************************
*         SYSTEM RESERVED AREAS NOT FOR MINIDISKS               *
*****************************************************************
*
USER $ALLOC$  NOLOG
 MDISK A01 3380 000 001 XASRES R
 MDISK B01 3380 000 001 XASERV R
 MDISK C01 3380 000 001 XAP001 R
*
USER $DIRECT$ NOLOG
 MDISK A04 3380 001 002 XASRES  R
*
USER $CP-NUC$ NOLOG
 MDISK A09 3380 003 010 XASRES  R
*
USER $SYSCKP$ NOLOG
 MDISK A06 3380 013 002 XASRES  R
*
USER $SYSWRM$ NOLOG
 MDISK A07 3380 015 002 XASRES  R
*
USER $PAGE$   NOLOG
 MDISK A03 3380 017 024 XASRES R
*
USER $SPOOL$  NOLOG
 MDISK B01 3380 041 087 XASRES R
*
USER $T-DISK$ NOLOG
 MDISK B01 3380 539 020 XASRES R
*
*****************************************************************
*             CMS USERIDS                                       *
*****************************************************************
*
USER CMS1  NOLOG  3M   32M G
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT4 CMSTST
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
*********************************************************************
*                  PROP LOGICAL OPERATOR                            *
*********************************************************************
*
USER LGLOPR  NOLOG 512K 16M ABCDEG
 ACCOUNT ACT1 PROP
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3380 493 001 XASRES WR
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
```

```
 LINK OPERATOR 191 291 RR
*
*****************************************************************
*             SYSTEM RELATED USERIDS                            *
*****************************************************************
*
USER MAINT NOLOG 16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
 IPL 190
 NAMESAVE GCS
 NAMESAVE VTAM
 NAMESAVE HELP
 NAMESAVE INSTHELP
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 MDISK 123 3380 000 885 XASRES  MW
 MDISK 124 3380 000 885 XAP001  RR
 MDISK 125 3380 000 885 XASERV  RR

 MDISK 191 3380 703 016 XASRES  MW
 MDISK 193 3380 128 046 XASRES  MW
 MDISK 194 3380 747 056 XASRES  MW
 MDISK 201 3380 043 022 XASERV  MW RMAINT   WMAINT  MMAINT

 MDISK 192 3380 065 015 XASERV  MW READ WRITE MULTIPLE
 MDISK 392 3380 080 015 XASERV  MW READ WRITE MULTIPLE
 MDISK 492 3380 095 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 692 3380 105 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 594 3380 115 060 XASERV  MW READ WRITE MULTIPLE
 MDISK 593 3380 175 040 XASERV  MW READ WRITE MULTIPLE

 MDISK 295 3380 559 030 XASRES  MW READ WRITE MULTIPLE
 MDISK 395 3380 589 020 XASRES  MW READ WRITE MULTIPLE
 MDISK 495 3380 265 005 XASERV  MW READ WRITE MULTIPLE
 MDISK 592 3380 270 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 892 3380 280 005 XASERV  MW READ WRITE MULTIPLE
 MDISK 491 3380 285 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 791 3380 295 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 89E 3380 305 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 896 3380 315 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 895 3380 325 010 XASERV  MW READ WRITE MULTIPLE

 MDISK 490 3380 362 072 XASERV  MW
 MDISK 423 3380 434 010 XASERV  MW
 MDISK 293 3380 444 060 XASERV  MW RCMSAUX  WCMSAUX MCMSAUX
 MDISK 294 3380 504 080 XASERV  MW RCPAUX   WCPAUX  MCPAUX
 MDISK 393 3380 389 056 XASRES  WR
 MDISK 394 3380 584 130 XASERV  WR
 MDISK 19C 3380 763 029 XASERV  WR
 MDISK 49C 3380 792 029 XASERV  WR

 MDISK 291 3380 803 060 XASRES  MW
 MDISK 391 3380 215 040 XASERV  MW
 MDISK 591 3380 821 060 XASERV  MW READ WRITE MULTIPLE
 MDISK 691 3380 306 040 XASRES  MW READ WRITE MULTIPLE
 MDISK 49D 3380 346 029 XASRES  MW READ WRITE MULTIPLE
 MDISK 501 3380 375 002 XASRES  MW READ WRITE MULTIPLE
```

```
 MDISK 595 3380 377 012 XASRES  MW READ WRITE MULTIPLE

 MDISK 190 3380 234 072 XASRES  MW ALL
 MDISK 19E 3380 659 044 XASRES  MW ALL
 MDISK 19D 3380 496 029 XASRES  MW ALL
 MDISK 596 3380 174 020 XASRES  MW ALL
 MDISK 59E 3380 525 010 XASRES  MW ALL
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
 MDISK 36E 3380 537 002 XASRES RR RPVM     WPVM     MPVM
 LINK RSCS  191 499  MW
*
USER CMSBATCH NOLOG 1M 2M G
 ACCOUNT 3 SYSTEM
 OPTION ACCT
 IPL CMS PARM AUTOCR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 195 3380 494 002 XASRES MR RBATCH   WBATCH   MBATCH
*
USER GCS NOLOG 16M 16M ABCDEFG
  AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT GCS RECVM
  IPL GCS
  NAMESAVE GCS
  CONSOLE 009 3215
  SPOOL 00C 2540 READER *
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 19D 19D RR
  LINK MAINT 595 595
  LINK MAINT 59E 59E
  MDISK 191 3380 463 004 XASRES MR RGCS WGCS MGCS
*
USER SYSMAINT NOLOG    3M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
  IPL CMS
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 123 123  MW
 LINK MAINT 191 192  RR
 LINK MAINT 190 190  RR
 LINK MAINT 19E 19E  RR
 LINK MAINT 19D 19D  RR
*
USER OPERATOR NOLOG   16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 2 OPERATOR
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
```

```
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 719 005 XASRES MR
 LINK OP1   191 192 RR
*
*OP1 IS AN ALTERNATE OPERATOR USERID
*
USER OP1 NOLOG 3M  32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 3 OPERATOR
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 724 001 XASRES MR
 LINK OPERATOR 191 192 RR
*
* This userid is for the IBM CE's use in running
* OLTSEP.
* OLTSEP is automatically IPLed in the virtual machine.
* A minimum machine size of one megabyte is required to
* run OLTSEP.
* The privilege class of F allows the CE to specify
* intensive recording mode.
* The console address of 01F is required by OLTSEP.
* The unit record addresses are those required by
* OLTSEP.
* The 5FF minidisk is the CE's OLTSEP pack.
*
USER OLTSEP NOLOG 3M 8M FG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT OLTSEP IBMCE
  IPL 5FF
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  MDISK 5FF 3380 000 885 CEPACK MR
*
*        **  EREP **
*
* This userid is for the IBM CE's use in running
* CPEREP.
* CMS is automatically IPLed in the virtual machine.
* The 190 minidisk is the CMS system disk.
* The 191 minidisk may be used to save often-used EREP
* control statements and procedures (EXECs).
* THE 190 MINIDISK HAS THE CMS TXTLIBS NECESSARY TO RUN
* EREP.
*
USER EREP NOLOG 3M 8M BFG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT EREP IBMCE
   IPL CMS
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
```

```
  SPOOL 00D 2540 PUNCH B
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 201 201 RR
  MDISK 191 3380 745 002 XASRES WR READ     WRITE   MULTIPLE
*
USER OPERATNS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 OPERATNS
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER D
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 725 017 XASRES MR RDVF     WDVF     MDVF
*
*
USER IPCS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 8 CE-ROOM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK OPERATNS 193 192 RR
*
USER AUTOLOG1  NOLOG   3M 8M ABCDEG
 AUTOLOG OP1 MAINT
 ACCOUNT 9 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 191 3380 742 001 XASRES MR RAUTOLOG WAUTOLOG MAUTOLOG
*
USER DISKACNT NOLOG 3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 743 001 XASRES MR READ WRITE MULTIPLE
*
USER IBMPSR NOLOG  3M 8M BG
 ACCOUNT 10 ACCNTNG
  IPL CMS
```

```
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 744 001 XASRES WR
*
USER IVPM2 NOLOG  3M   4M G
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER VMUTIL NOLOG  3M 8M BDEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 11 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER SYSDUMP1 NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 12 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 123 3380 000 885 XASRES RR
 MDISK 124 3380 000 885 XAP001 RR
 MDISK 125 3380 000 885 XASERV RR
*
*
USER RSCS     NOLOG   3M 8M BEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 13 SYSTEM
  NOPDATA
  IPL 191
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
```

```
 LINK MAINT 19D 19D RR
 MDISK 191 3380 535 002 XASRES MR RRSCS    WRSCS    MRSCS
*
USER PVM      NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 14 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK MAINT 36E 191 MR
*
*
*****************************************************************
*             OTHER OPERATING SYSTEM USERIDS                    *
*****************************************************************
*
```

Sample XAMAINT Directory Entry for 3380

If you are using the procedure in Chapter 4, "Installing VM/XA System Product Release 2 Using an Existing VM/SP or VM/SP HPO System" on page 219 to install your system, add the following entry to your directory:

```
*
USER XAMAINT NOLOG 16M 16M ABCDEFG
 OPTION ECMODE DIAG98
 ACCOUNT 1 SYSPROG
 IPL 190
*NAMESAVE GCS
*NAMESAVE VTAM
*NAMESAVE HELP
*NAMESAVE INSTHELP
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 MDISK 123 3380 000 885 XASRES  MW
 MDISK 124 3380 000 885 XAP001  RR
 MDISK 125 3380 000 885 XASERV  RR

 MDISK 191 3380 703 016 XASRES  MW
 MDISK 193 3380 128 046 XASRES  MW
 MDISK 194 3380 747 056 XASRES  MW
 MDISK 201 3380 043 022 XASERV  MW RMAINT   WMAINT  MMAINT

 MDISK 192 3380 065 015 XASERV  MW READ WRITE MULTIPLE
 MDISK 392 3380 080 015 XASERV  MW READ WRITE MULTIPLE
 MDISK 492 3380 095 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 692 3380 105 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 594 3380 115 060 XASERV  MW READ WRITE MULTIPLE
 MDISK 593 3380 175 040 XASERV  MW READ WRITE MULTIPLE

 MDISK 295 3380 559 030 XASRES  MW READ WRITE MULTIPLE
 MDISK 395 3380 589 020 XASRES  MW READ WRITE MULTIPLE
 MDISK 495 3380 265 005 XASERV  MW READ WRITE MULTIPLE
 MDISK 592 3380 270 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 892 3380 280 005 XASERV  MW READ WRITE MULTIPLE
 MDISK 491 3380 285 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 791 3380 295 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 89E 3380 305 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 896 3380 315 010 XASERV  MW READ WRITE MULTIPLE
 MDISK 895 3380 325 010 XASERV  MW READ WRITE MULTIPLE

 MDISK 490 3380 362 072 XASERV  MW
 MDISK 423 3380 434 010 XASERV  MW
 MDISK 293 3380 444 060 XASERV  MW RCMSAUX  WCMSAUX MCMSAUX
 MDISK 294 3380 504 080 XASERV  MW RCPAUX   WCPAUX  MCPAUX
 MDISK 393 3380 389 056 XASRES  WR
 MDISK 394 3380 584 130 XASERV  WR
 MDISK 19C 3380 763 029 XASERV  WR
 MDISK 49C 3380 792 029 XASERV  WR

 MDISK 291 3380 803 060 XASRES  MW
 MDISK 391 3380 215 040 XASERV  MW
 MDISK 591 3380 821 060 XASERV  MW READ WRITE MULTIPLE
 MDISK 691 3380 306 040 XASRES  MW READ WRITE MULTIPLE
 MDISK 49D 3380 346 029 XASRES  MW READ WRITE MULTIPLE
```

```
 MDISK 501 3380 375 002 XASRES  MW READ WRITE MULTIPLE
 MDISK 595 3380 377 012 XASRES  MW READ WRITE MULTIPLE

 MDISK 190 3380 234 072 XASRES  MW ALL
 MDISK 19E 3380 659 044 XASRES  MW ALL
 MDISK 19D 3380 496 029 XASRES  MW ALL
 MDISK 596 3380 174 020 XASRES  MW ALL
 MDISK 59E 3380 525 010 XASRES  MW ALL
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
 MDISK 36E 3380 537 002 XASRES RR RPVM     WPVM     MPVM
 LINK RSCS  191 499  MW
 LINK MAINT 190 390  RR
*
```

System Residence DASD Allocation for 3380

Figure 58 shows the allocation for the system residence device for the 3380 VM/XA SP starter system. The allocation and minidisk maps for the 3380-E4 ("System Residence DASD Allocation for 3380-E4" on page 828 and "Minidisk Maps for 3380-E4 System Residence Device" on page 829) and 3380-K ("System Residence DASD Allocation for 3380-K" on page 843 and "Minidisk Maps for 3380-K System Residence Device" on page 844) follow the corresponding sample directories ("Sample Directory for 3380-E4" on page 817 and "Sample Directory for 3380-K" on page 832 respectively).

```
TYPE CYL CYL
PERM 000-000
DRCT 001-002
PERM 003-013
PAGE 014-037
SPOL 038-074
PERM 075-197
TDSK 198-216
PERM 217-884
```

Figure 58. System Residence DASD Allocation for 3380

Minidisk Maps for 3380 System Residence Device

Figure 59 shows the minidisk maps of the system residence device for the 3380. To map your own system residence device for comparison, issue DISKMAP *fn* DIRECT, where *fn* is the name of your directory.

```
------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END      SIZE
CEPACK   OLTSEP      5FF     3380     0000     0884      0885

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END      SIZE
XAP001   $ALLOC$     C01     3380     0000     0000      0001
         MAINT       124     3380     0000     0884      0885 **OVERLAP**
         SYSDUMP1    124     3380     0000     0884      0885 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END      SIZE
XASERV   $ALLOC$     B01     3380     0000     0000      0001
                                         1       42        42    GAP
         MAINT       201     3380     0043     0064      0022
         MAINT       192     3380     0065     0079      0015
         MAINT       392     3380     0080     0094      0015
         MAINT       492     3380     0095     0104      0010
         MAINT       692     3380     0105     0114      0010
         MAINT       594     3380     0115     0174      0060
         MAINT       593     3380     0175     0214      0040
         MAINT       391     3380     0215     0254      0040
                                        255      264        10    GAP
         MAINT       495     3380     0265     0269      0005
         MAINT       592     3380     0270     0279      0010
         MAINT       892     3380     0280     0284      0005
         MAINT       491     3380     0285     0294      0010
         MAINT       791     3380     0295     0304      0010
         MAINT       89E     3380     0305     0314      0010
         MAINT       896     3380     0315     0324      0010
         MAINT       895     3380     0325     0334      0010
                                        335      361        27    GAP
         MAINT       490     3380     0362     0433      0072
         MAINT       423     3380     0434     0443      0010
         MAINT       293     3380     0444     0503      0060
         MAINT       294     3380     0504     0583      0080
         MAINT       394     3380     0584     0713      0130
                                        714      762        49    GAP
```

Figure 59 (Part 1 of 3). Minidisk Maps of the System Residence Device for 3380

```
         MAINT      19C     3380     0763     0791     0029
         MAINT      49C     3380     0792     0820     0029
         MAINT      591     3380     0821     0880     0060
         MAINT      125     3380     0000     0884     0885 **OVERLAP**
         SYSDUMP1   125     3380     0000     0884     0885 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID     CUU   DEVTYPE   START      END      SIZE
XASRES   $ALLOC$    A01     3380     0000     0000     0001
         $DIRECT$   A04     3380     0001     0002     0002
         $CP-NUC$   A09     3380     0003     0012     0010
         $SYSCKP$   A06     3380     0013     0014     0002
         $SYSWRM$   A07     3380     0015     0016     0002
         $PAGE$     A03     3380     0017     0040     0024
         $SPOOL$    B01     3380     0041     0127     0087
         MAINT      193     3380     0128     0173     0046
         MAINT      596     3380     0174     0193     0020
                                      194      233       40    GAP
         MAINT      190     3380     0234     0305     0072
         MAINT      691     3380     0306     0345     0040
         MAINT      49D     3380     0346     0374     0029
         MAINT      501     3380     0375     0376     0002
         MAINT      595     3380     0377     0388     0012
         MAINT      393     3380     0389     0444     0056
                                      445      462       18    GAP
         GCS        191     3380     0463     0466     0004
                                      467      492       26    GAP
         LGLOPR     191     3380     0493     0493     0001
         CMSBATCH   195     3380     0494     0495     0002
         MAINT      19D     3380     0496     0524     0029
         MAINT      59E     3380     0525     0534     0010
         RSCS       191     3380     0535     0536     0002
         MAINT      36E     3380     0537     0538     0002
         $T-DISK$   B01     3380     0539     0558     0020
         MAINT      295     3380     0559     0588     0030
         MAINT      395     3380     0589     0608     0020
                                      609      658       50    GAP
```

Figure 59 (Part 2 of 3). Minidisk Maps of the System Residence Device for 3380

```
         MAINT      19E     3380     0659     0702     0044
         MAINT      191     3380     0703     0718     0016
         OPERATOR   191     3380     0719     0723     0005
         OP1        191     3380     0724     0724     0001
         OPERATNS   191     3380     0725     0741     0017
         AUTOLOG1   191     3380     0742     0742     0001
         DISKACNT   191     3380     0743     0743     0001
         IBMPSR     191     3380     0744     0744     0001
         EREP       191     3380     0745     0746     0002
         MAINT      194     3380     0747     0802     0056
         MAINT      291     3380     0803     0862     0060
         MAINT      123     3380     0000     0884     0885 **OVERLAP**
         SYSDUMP1   123     3380     0000     0884     0885 **OVERLAP**
```

Figure 59 (Part 3 of 3). Minidisk Maps of the System Residence Device for 3380

Sample Directory for 3380-E4

This is only a sample file. You need to tailor it to your installation before it can be used.

```
************************************************************
***     Virtual Machine / System Product       5664-308    ***
***     Contains restricted materials of IBM               ***
***     Copyright (c) I B M   Corporation          1988    ***
***     Licensed Materials -  Property of I B M            ***
***     Refer to Copyright Instructions: Form G120-2083    ***
************************************************************
************************************************************
*  3380-E4 SYSTEM VM/XA SYSTEM  PRODUCT  DIRECTORY          *
*  THE ADDRESSES 123, 124, AND 125 ARE VIRTUAL ADDRESSES.   *
*  THE ADDRESS 123 IS CRITICAL SINCE IT IS USED IN HCPSYS AND *
*  THE DIRECTORY. IF YOU WANT TO CHANGE IT, REMEMBER IT MUST  *
*  BE CHANGED IN HCPSYS, THE 'DIRECTORY' STATEMENT BELOW,     *
*  AND IN THE 'MDISK' STATEMENTS FOR THE USERID  'MAINT'.     *
*  NOTE: REMEMBER THESE ARE ONLY VIRTUAL ADDRESSES NOT REAL   *
*  ADDRESSES, SO THERE IS NO NEED TO CHANGE THEM TO MATCH     *
*  YOUR HARDWARE ADDRESSES.                                   *
*                                                             *
*  THIS DIRECTORY MUST BE MODIFIED IF MDISK PASSWORDS ARE TO  *
*  BE SPECIFIED.  WITHOUT MDISK PASSWORDS, MDISKS CANNOT BE   *
*  SHARED USING THE 'CP LINK' COMMAND.  A SAMPLE USER ENTRY   *
*  FOLLOWS: (NOTE ONLY THE MDISK STATEMENT)                   *
*                                                             *
*      USER ANYONE SOMEPASS 3M 8M FG                          *
*       AUTOLOG AUTOLOG1 OP1 MAINT                            *
*        ACCOUNT ??????                                       *
*        IPL 190                                              *
*        CONSOLE 01F 3215                                     *
*        SPOOL 00C 2540 READER A                              *
*        SPOOL 00D 2540 PUNCH A                               *
*        SPOOL 00E 1403 A                                     *
*====>   MDISK 999 33XX XXX YYY ZZZZZZ MR RPASS WPASS MPASS   *
*                                                             *
*    WHERE:  RPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                READ ONLY MODE.                              *
*            WPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                WRITE MODE.                                  *
*            MPASS WILL BE THE REQUIRED PASSWORD TO LINK IN   *
*                MULTI-WRITE MODE.                            *
*                                                             *
*   NOTE:  EACH PASSWORD MAY BE 1-8 CHARACTERS IN LENGTH.     *
*                                                             *
************************************************************
*                                                            *
*                                                            *
*  Following is a sample user entry and a brief description  *
*  of each entry.  A more detailed description may be found  *
*  by referencing the VM/XA System Product Planning and      *
*  Administration manual.                                    *
*                                                            *
*    USER CMS1  abc123  3M  32M G                            *
*       -  Userid is CMS1                                    *
*       -  Logon password is abc123                          *
*       -  At logon time, virtual storage will equal 3M      *
*       -  Maximum storage that may be defined equals 32M    *
*       -  Privilege class is G                              *
```

```
*                                                                     *
*     AUTOLOG AUTOLOG1 OP1 MAINT                                      *
*       - Allows user CMS1 to be autolog'd by AUTOLOG1,               *
*         OP1, and MAINT                                              *
*                                                                     *
*     ACCOUNT ACT4 CMSTST                                             *
*       - Time used by CMS1 will be charged to account ACT4           *
*       - Printed output will be sent to distribution CMSTST          *
*                                                                     *
*     IPL CMS                                                         *
*       - Will cause automatic IPL of named save system of            *
*         CMS                                                         *
*                                                                     *
*     CONSOLE 009 3215                                                *
*       - Defines virtual machines console as a 3215 at a             *
*         virtual address of 009                                      *
*                                                                     *
*     SPOOL 00C 2540 READER A                                         *
*     SPOOL 00D 2540 PUNCH A                                          *
*     SPOOL 00E 1403 A                                                *
*       - Defines virtual unit record devices of reader,              *
*         punch, and printer with a spooling class of 'A'             *
*                                                                     *
*     LINK MAINT 190 190 RR                                           *
*       - Allows read access to minidisk 190 of user 'MAINT'          *
*                                                                     *
*     LINK MAINT 19E 19E RR                                           *
*       - Allows read access to minidisk 19E of user 'MAINT'          *
*                                                                     *
*     LINK MAINT 19D 19D RR                                           *
*       - Allows read access to minidisk 19d of user 'MAINT'          *
*                                                                     *
*     MDISK 999 3350 236 022 CPPACK WR RPASS WPASS                    *
*       - Defines minidisk with a virtual address of 999              *
*       - Specifies that the minidisk resides on a real 3350          *
*       - Minidisk starts at cylinder 236                             *
*       - Mdisk is 22 cylinders in size (starting at cyl 236          *
*       - The real volume serial number is 'CPPACK'                   *
*       - Mdisk is to be accessed in write mode if no other           *
*         user has write access. Alternate access read-only           *
*       - RPASS is the required password for another user to          *
*         link to this minidisk in read mode                          *
*       - WPASS is the required password for another user to          *
*         link to this minidisk in write mode                         *
***********************************************************************
*
*
*
DIRECTORY 123 3380 XASRES
*
***********************************************************************
*         SYSTEM RESERVED AREAS NOT FOR MINIDISKS                     *
***********************************************************************
*
USER $ALLOC$  NOLOG
 MDISK A01 3380 000 001 XASRES R
 MDISK B01 3380 000 001 XASERV R
 MDISK C01 3380 000 001 XAP001 R
*
USER $DIRECT$ NOLOG
```

```
 MDISK A04 3380 001 002 XASRES  R
*
USER $CP-NUC$ NOLOG
 MDISK A09 3380 003 010 XASRES  R
*
USER $SYSCKP$ NOLOG
 MDISK A06 3380 013 002 XASRES  R
*
USER $SYSWRM$ NOLOG
 MDISK A07 3380 015 002 XASRES  R
*
USER $PAGE$   NOLOG
 MDISK A03 3380 017 024 XASRES R
*
USER $SPOOL$  NOLOG
 MDISK B01 3380 041 087 XASRES R
*
USER $T-DISK$ NOLOG
 MDISK B01 3380 539 020 XASRES R
*
*****************************************************************
*             CMS USERIDS                                       *
*****************************************************************
*
USER CMS1  NOLOG  3M   32M G
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT4 CMSTST
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
*********************************************************************
*                  PROP LOGICAL OPERATOR                            *
*********************************************************************
*
USER LGLOPR  NOLOG 512K 16M ABCDEG
 ACCOUNT ACT1 PROP
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3380 493 001 XASRES WR
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK OPERATOR 191 291 RR
*
*****************************************************************
*             SYSTEM RELATED USERIDS                            *
*****************************************************************
*
USER MAINT NOLOG 16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
 IPL 190
```

```
NAMESAVE GCS
NAMESAVE VTAM
NAMESAVE HELP
NAMESAVE INSTHELP
CONSOLE 009 3215 T
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH  A
SPOOL 00E 1403 A
MDISK 123 3380 0000 1770 XASRES  MW
MDISK 124 3380 0000 1770 XAP001  RR
MDISK 125 3380 0000 1770 XASERV  RR

MDISK 191 3380 0703 016 XASRES  MW
MDISK 193 3380 0128 046 XASRES  MW
MDISK 194 3380 0362 056 XASRES  MW
MDISK 201 3380 0789 022 XASRES  MW RMAINT   WMAINT   MMAINT

MDISK 192 3380 0811 015 XASRES  MW READ WRITE MULTIPLE
MDISK 392 3380 0826 015 XASRES  MW READ WRITE MULTIPLE
MDISK 492 3380 0841 010 XASRES  MW READ WRITE MULTIPLE
MDISK 692 3380 0851 010 XASRES  MW READ WRITE MULTIPLE

MDISK 593 3380 0861 040 XASRES  MW READ WRITE MULTIPLE
MDISK 594 3380 0901 060 XASRES  MW READ WRITE MULTIPLE

MDISK 295 3380 0961 030 XASRES  MW READ WRITE MULTIPLE
MDISK 395 3380 0991 020 XASRES  MW READ WRITE MULTIPLE
MDISK 495 3380 1011 005 XASRES  MW READ WRITE MULTIPLE
MDISK 592 3380 1016 010 XASRES  MW READ WRITE MULTIPLE
MDISK 892 3380 1026 005 XASRES  MW READ WRITE MULTIPLE
MDISK 491 3380 1031 010 XASRES  MW READ WRITE MULTIPLE
MDISK 791 3380 1041 010 XASRES  MW READ WRITE MULTIPLE
MDISK 89E 3380 1051 010 XASRES  MW READ WRITE MULTIPLE
MDISK 896 3380 1061 010 XASRES  MW READ WRITE MULTIPLE
MDISK 895 3380 1071 010 XASRES  MW READ WRITE MULTIPLE

MDISK 490 3380 1108 072 XASRES  MW
MDISK 423 3380 1180 010 XASRES  MW
MDISK 293 3380 1190 060 XASRES  MW RCMSAUX  WCMSAUX  MCMSAUX
MDISK 294 3380 1250 080 XASRES  MW RCPAUX   WCPAUX   MCPAUX
MDISK 393 3380 0306 056 XASRES  WR
MDISK 394 3380 1330 130 XASRES  WR
MDISK 19C 3380 1509 029 XASRES  WR
MDISK 49C 3380 1538 029 XASRES  WR
MDISK 596 3380 1567 020 XASRES  WR
MDISK 59E 3380 1587 010 XASRES  WR

MDISK 391 3380 0559 040 XASRES  MW
MDISK 591 3380 1597 060 XASRES  MW READ WRITE MULTIPLE
MDISK 691 3380 1657 040 XASRES  MW READ WRITE MULTIPLE
MDISK 49D 3380 1697 029 XASRES  MW READ WRITE MULTIPLE
MDISK 501 3380 1726 002 XASRES  MW READ WRITE MULTIPLE
MDISK 595 3380 1728 012 XASRES  MW READ WRITE MULTIPLE
MDISK 291 3380 0599 060 XASRES  MW

MDISK 190 3380 0234 072 XASRES  MW ALL
MDISK 19E 3380 0174 044 XASRES  MW ALL
MDISK 19D 3380 0659 029 XASRES  MW ALL
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
```

```
 MDISK 36E 3380 0537 002 XASRES RR RPVM     WPVM     MPVM
 LINK RSCS  191 499  MW
*
USER CMSBATCH NOLOG 1M 2M G
 ACCOUNT 3 SYSTEM
 OPTION ACCT
 IPL CMS PARM AUTOCR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 195 3380 494 002 XASRES MR RBATCH   WBATCH   MBATCH
*
USER GCS NOLOG 16M 16M ABCDEFG
  AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT GCS RECVM
  IPL GCS
  NAMESAVE GCS
  CONSOLE 009 3215
  SPOOL 00C 2540 READER *
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 19D 19D RR
  LINK MAINT 595 595
  LINK MAINT 59E 59E
  MDISK 191 3380 463 004 XASRES MR RGCS WGCS MGCS
*
USER SYSMAINT NOLOG    3M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
  IPL CMS
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 123 123  MW
 LINK MAINT 191 192  RR
 LINK MAINT 190 190  RR
 LINK MAINT 19E 19E  RR
 LINK MAINT 19D 19D  RR
*
USER OPERATOR NOLOG   16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 2 OPERATOR
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 719 005 XASRES MR
 LINK OP1   191 192 RR
*
*OP1 IS AN ALTERNATE OPERATOR USERID
*
USER OP1 NOLOG 3M  32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
```

```
 ACCOUNT 3 OPERATOR
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 724 001 XASRES MR
 LINK OPERATOR 191 192 RR
*
* This userid is for the IBM CE's use in running
* OLTSEP.
* OLTSEP is automatically IPLed in the virtual machine.
* A minimum machine size of one megabyte is required to
* run OLTSEP.
* The privilege class of F allows the CE to specify
* intensive recording mode.
* The console address of 01F is required by OLTSEP.
* The unit record addresses are those required by
* OLTSEP.
* The 5FF minidisk is the CE's OLTSEP pack.
*
USER OLTSEP NOLOG 3M 8M FG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT OLTSEP IBMCE
  IPL 5FF
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  MDISK 5FF 3380 000 1770 CEPACK MR
*
*        **  EREP **
*
* This userid is for the IBM CE's use in running
* CPEREP.
* CMS is automatically IPLed in the virtual machine.
* The 190 minidisk is the CMS system disk.
* The 191 minidisk may be used to save often-used EREP
* control statements and procedures (EXECs).
* THE 190 MINIDISK HAS THE CMS TXTLIBS NECESSARY TO RUN
* EREP.
*
USER EREP NOLOG 3M 8M BFG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT EREP IBMCE
   IPL CMS
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH B
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 201 201 RR
  MDISK 191 3380 745 002 XASRES WR READ     WRITE    MULTIPLE
*
USER OPERATNS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 OPERATNS
```

```
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER D
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 725 017 XASRES MR RDVF     WDVF     MDVF
*
*
USER IPCS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 8 CE-ROOM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK OPERATNS 193 192 RR
*
USER AUTOLOG1  NOLOG   3M 8M ABCDEG
 AUTOLOG OP1 MAINT
 ACCOUNT 9 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 191 3380 742 001 XASRES MR RAUTOLOG WAUTOLOG MAUTOLOG
*
USER DISKACNT NOLOG 3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 743 001 XASRES MR READ WRITE MULTIPLE
*
USER IBMPSR NOLOG  3M 8M BG
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
```

```
 MDISK 191 3380 744 001 XASRES WR
*
USER IVPM2 NOLOG  3M   4M G
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER VMUTIL NOLOG  3M 8M BDEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 11 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER SYSDUMP1 NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 12 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 123 3380 000 1770 XASRES RR
 MDISK 124 3380 000 1770 XAP001 RR
 MDISK 125 3380 000 1770 XASERV RR
*
*
USER RSCS     NOLOG   3M 8M BEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 13 SYSTEM
  NOPDATA
  IPL 191
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 535 002 XASRES MR RRSCS    WRSCS    MRSCS
*
USER PVM      NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 14 SYSTEM
  IPL CMS
```

```
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK MAINT 36E 191 MR
*
*
*****************************************************************
*             OTHER OPERATING SYSTEM USERIDS                    *
*****************************************************************
*
```

Sample XAMAINT Directory Entry for 3380-E4

If you are using the procedure in Chapter 4, "Installing VM/XA System Product Release 2 Using an Existing VM/SP or VM/SP HPO System" on page 219 to install your system, add the following entry to your directory:

```
*
USER XAMAINT NOLOG 16M 16M ABCDEFG
 OPTION ECMODE DIAG98
 ACCOUNT 1 SYSPROG
 IPL 190
*NAMESAVE GCS
*NAMESAVE VTAM
*NAMESAVE HELP
*NAMESAVE INSTHELP
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 MDISK 123 3380 0000 1770 XASRES  MW
 MDISK 124 3380 0000 1770 XAP001  RR
 MDISK 125 3380 0000 1770 XASERV  RR

 MDISK 191 3380 0703 016 XASRES  MW
 MDISK 193 3380 0128 046 XASRES  MW
 MDISK 194 3380 0362 056 XASRES  MW
 MDISK 201 3380 0789 022 XASRES  MW RMAINT   WMAINT   MMAINT

 MDISK 192 3380 0811 015 XASRES  MW READ WRITE MULTIPLE
 MDISK 392 3380 0826 015 XASRES  MW READ WRITE MULTIPLE
 MDISK 492 3380 0841 010 XASRES  MW READ WRITE MULTIPLE
 MDISK 692 3380 0851 010 XASRES  MW READ WRITE MULTIPLE

 MDISK 593 3380 0861 040 XASRES  MW READ WRITE MULTIPLE
 MDISK 594 3380 0901 060 XASRES  MW READ WRITE MULTIPLE

 MDISK 295 3380 0961 030 XASRES  MW READ WRITE MULTIPLE
 MDISK 395 3380 0991 020 XASRES  MW READ WRITE MULTIPLE
 MDISK 495 3380 1011 005 XASRES  MW READ WRITE MULTIPLE
 MDISK 592 3380 1016 010 XASRES  MW READ WRITE MULTIPLE
 MDISK 892 3380 1026 005 XASRES  MW READ WRITE MULTIPLE
 MDISK 491 3380 1031 010 XASRES  MW READ WRITE MULTIPLE
 MDISK 791 3380 1041 010 XASRES  MW READ WRITE MULTIPLE
 MDISK 89E 3380 1051 010 XASRES  MW READ WRITE MULTIPLE
 MDISK 896 3380 1061 010 XASRES  MW READ WRITE MULTIPLE
 MDISK 895 3380 1071 010 XASRES  MW READ WRITE MULTIPLE

 MDISK 490 3380 1108 072 XASRES  MW
 MDISK 423 3380 1180 010 XASRES  MW
 MDISK 293 3380 1190 060 XASRES  MW RCMSAUX  WCMSAUX  MCMSAUX
 MDISK 294 3380 1250 080 XASRES  MW RCPAUX   WCPAUX   MCPAUX
 MDISK 393 3380 0306 056 XASRES  WR
 MDISK 394 3380 1330 130 XASRES  WR
 MDISK 19C 3380 1509 029 XASRES  WR
 MDISK 49C 3380 1538 029 XASRES  WR
 MDISK 596 3380 1567 020 XASRES  WR
 MDISK 59E 3380 1587 010 XASRES  WR

 MDISK 391 3380 0559 040 XASRES  MW
 MDISK 591 3380 1597 060 XASRES  MW READ WRITE MULTIPLE
```

```
 MDISK 691 3380 1657 040 XASRES  MW READ WRITE MULTIPLE
 MDISK 49D 3380 1697 029 XASRES  MW READ WRITE MULTIPLE
 MDISK 501 3380 1726 002 XASRES  MW READ WRITE MULTIPLE
 MDISK 595 3380 1728 012 XASRES  MW READ WRITE MULTIPLE
 MDISK 291 3380 0599 060 XASRES  MW

 MDISK 190 3380 0234 072 XASRES  MW ALL
 MDISK 19E 3380 0174 044 XASRES  MW ALL
 MDISK 19D 3380 0659 029 XASRES  MW ALL
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
 MDISK 36E 3380 0537 002 XASRES RR RPVM     WPVM     MPVM
 LINK RSCS  191 499  MW
 LINK MAINT 190 390  RR
*
```

System Residence DASD Allocation for 3380-E4

Figure 60 shows the allocation for the system residence device for the 3380-E4 VM/XA SP Starter System.

```
TYPE  CYL  CYL
PERM 0000-0000
DRCT 0001-0002
PERM 0003-0013
PAGE 0014-0037
SPOL 0038-0074
PERM 0075-0197
TDSK 0198-0216
PERM 0217-1769
```

Figure 60. System Residence DASD Allocation for 3380-E4

Minidisk Maps for 3380-E4 System Residence Device

Figure 61 shows the minidisk maps of the system residence device for the 3380-E4. To map your own system residence device for comparison, issue DISKMAP *fn* DIRECT, where *fn* is the name of your directory.

```
------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
CEPACK   OLTSEP      5FF     3380     0000     1769     1770

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XAP001   $ALLOC$     C01     3380     0000     0000     0001
         MAINT       124     3380     0000     1769     1770 **OVERLAP**
         SYSDUMP1    124     3380     0000     1769     1770 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XASERV   $ALLOC$     B01     3380     0000     0000     0001
         MAINT       125     3380     0000     1769     1770 **OVERLAP**
         SYSDUMP1    125     3380     0000     1769     1770 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XASRES   $ALLOC$     A01     3380     0000     0000     0001
         $DIRECT$    A04     3380     0001     0002     0002
         $CP-NUC$    A09     3380     0003     0012     0010
         $SYSCKP$    A06     3380     0013     0014     0002
         $SYSWRM$    A07     3380     0015     0016     0002
         $PAGE$      A03     3380     0017     0040     0024
         $SPOOL$     B01     3380     0041     0127     0087
         MAINT       193     3380     0128     0173     0046
         MAINT       19E     3380     0174     0217     0044
                                       218      233       16    GAP
         MAINT       190     3380     0234     0305     0072
         MAINT       393     3380     0306     0361     0056
         MAINT       194     3380     0362     0417     0056
                                       418      462       45    GAP
         GCS         191     3380     0463     0466     0004
                                       467      492       26    GAP
```

Figure 61 (Part 1 of 3). Minidisk Maps of the System Residence Device for 3380-E4

LGLOPR	191	3380	0493	0493	0001	
CMSBATCH	195	3380	0494	0495	0002	
			496	534	39	GAP
RSCS	191	3380	0535	0536	0002	
MAINT	36E	3380	0537	0538	0002	
$T-DISK$	B01	3380	0539	0558	0020	
MAINT	391	3380	0559	0598	0040	
MAINT	291	3380	0599	0658	0060	
MAINT	19D	3380	0659	0687	0029	
			688	702	15	GAP
MAINT	191	3380	0703	0718	0016	
OPERATOR	191	3380	0719	0723	0005	
OP1	191	3380	0724	0724	0001	
OPERATNS	191	3380	0725	0741	0017	
AUTOLOG1	191	3380	0742	0742	0001	
DISKACNT	191	3380	0743	0743	0001	
IBMPSR	191	3380	0744	0744	0001	
EREP	191	3380	0745	0746	0002	
			747	788	42	GAP
MAINT	201	3380	0789	0810	0022	
MAINT	192	3380	0811	0825	0015	
MAINT	392	3380	0826	0840	0015	
MAINT	492	3380	0841	0850	0010	
MAINT	692	3380	0851	0860	0010	
MAINT	593	3380	0861	0900	0040	
MAINT	594	3380	0901	0960	0060	
MAINT	295	3380	0961	0990	0030	
MAINT	395	3380	0991	1010	0020	
MAINT	495	3380	1011	1015	0005	
MAINT	592	3380	1016	1025	0010	
MAINT	892	3380	1026	1030	0005	
MAINT	491	3380	1031	1040	0010	
MAINT	791	3380	1041	1050	0010	
MAINT	89E	3380	1051	1060	0010	
MAINT	896	3380	1061	1070	0010	
MAINT	895	3380	1071	1080	0010	
			1081	1107	27	GAP

Figure 61 (Part 2 of 3). Minidisk Maps of the System Residence Device for 3380-E4

```
MAINT     490    3380    1108    1179    0072
MAINT     423    3380    1180    1189    0010
MAINT     293    3380    1190    1249    0060
MAINT     294    3380    1250    1329    0080
MAINT     394    3380    1330    1459    0130
                         1460    1508      49    GAP
MAINT     19C    3380    1509    1537    0029
MAINT     49C    3380    1538    1566    0029
MAINT     596    3380    1567    1586    0020
MAINT     59E    3380    1587    1596    0010
MAINT     591    3380    1597    1656    0060
MAINT     691    3380    1657    1696    0040
MAINT     49D    3380    1697    1725    0029
MAINT     501    3380    1726    1727    0002
MAINT     595    3380    1728    1739    0012
MAINT     123    3380    0000    1769    1770 **OVERLAP**
SYSDUMP1  123    3380    0000    1769    1770 **OVERLAP**
```

Figure 61 (Part 3 of 3). Minidisk Maps of the System Residence Device for 3380-E4

Sample Directory for 3380-K

This is only a sample file. You need to tailor it to your installation before it can be used.

```
*************************************************************
***     Virtual Machine / System Product       5664-308    ***
***     Contains restricted materials of IBM               ***
***     Copyright (c) I B M   Corporation          1988    ***
***     Licensed Materials -  Property of I B M            ***
***     Refer to Copyright Instructions: Form G120-2083    ***
*************************************************************
*************************************************************
*  3380-3 SYSTEM VM/XA SYSTEM  PRODUCT  DIRECTORY           *
*  THE ADDRESSES 123, 124, AND 125 ARE VIRTUAL ADDRESSES.   *
*  THE ADDRESS 123 IS CRITICAL SINCE IT IS USED IN HCPSYS AND *
*  THE DIRECTORY. IF YOU WANT TO CHANGE IT, REMEMBER IT MUST  *
*  BE CHANGED IN HCPSYS, THE 'DIRECTORY' STATEMENT BELOW,   *
*  AND IN THE 'MDISK' STATEMENTS FOR THE USERID  'MAINT'.   *
*  NOTE: REMEMBER THESE ARE ONLY VIRTUAL ADDRESSES NOT REAL   *
*  ADDRESSES, SO THERE IS NO NEED TO CHANGE THEM TO MATCH    *
*  YOUR HARDWARE ADDRESSES.                                 *
*                                                           *
*  THIS DIRECTORY MUST BE MODIFIED IF MDISK PASSWORDS ARE TO  *
*  BE SPECIFIED.  WITHOUT MDISK PASSWORDS, MDISKS CANNOT BE   *
*  SHARED USING THE 'CP LINK' COMMAND.  A SAMPLE USER ENTRY   *
*  FOLLOWS: (NOTE ONLY THE MDISK STATEMENT)                 *
*                                                           *
*      USER ANYONE SOMEPASS 3M 8M FG                        *
*       AUTOLOG AUTOLOG1 OP1 MAINT                          *
*        ACCOUNT ??????                                     *
*        IPL 190                                            *
*        CONSOLE 01F 3215                                   *
*        SPOOL 00C 2540 READER A                            *
*        SPOOL 00D 2540 PUNCH A                             *
*        SPOOL 00E 1403 A                                   *
*====>   MDISK 999 33XX XXX YYY ZZZZZZ MR RPASS WPASS MPASS  *
*                                                           *
*    WHERE:  RPASS WILL BE THE REQUIRED PASSWORD TO LINK IN  *
*                READ ONLY MODE.                            *
*            WPASS WILL BE THE REQUIRED PASSWORD TO LINK IN  *
*                WRITE MODE.                                *
*            MPASS WILL BE THE REQUIRED PASSWORD TO LINK IN  *
*                MULTI-WRITE MODE.                          *
*                                                           *
*   NOTE:  EACH PASSWORD MAY BE 1-8 CHARACTERS IN LENGTH.   *
*                                                           *
*************************************************************
*                                                           *
*                                                           *
*  Following is a sample user entry and a brief description  *
*  of each entry.  A more detailed description may be found  *
*  by referencing the VM/XA System Product Planning and     *
*  Administration manual.                                   *
*                                                           *
*    USER CMS1  abc123  3M  32M G                           *
*       -  Userid is CMS1                                   *
*       -  Logon password is abc123                         *
*       -  At logon time, virtual storage will equal 3M     *
*       -  Maximum storage that may be defined equals 32M   *
*       -  Privilege class is G                             *
```

```
*                                                                     *
*     AUTOLOG AUTOLOG1 OP1 MAINT                                      *
*       - Allows user CMS1 to be autolog'd by AUTOLOG1,               *
*         OP1, and MAINT                                              *
*                                                                     *
*     ACCOUNT ACT4 CMSTST                                             *
*       - Time used by CMS1 will be charged to account ACT4           *
*       - Printed output will be sent to distribution CMSTST          *
*                                                                     *
*     IPL CMS                                                         *
*       - Will cause automatic IPL of named save system of            *
*         CMS                                                         *
*                                                                     *
*     CONSOLE 009 3215                                                *
*       - Defines virtual machines console as a 3215 at a             *
*         virtual address of 009                                      *
*                                                                     *
*     SPOOL 00C 2540 READER A                                         *
*     SPOOL 00D 2540 PUNCH A                                          *
*     SPOOL 00E 1403 A                                                *
*       - Defines virtual unit record devices of reader,              *
*         punch, and printer with a spooling class of 'A'             *
*                                                                     *
*     LINK MAINT 190 190 RR                                           *
*       - Allows read access to minidisk 190 of user 'MAINT'          *
*                                                                     *
*     LINK MAINT 19E 19E RR                                           *
*       - Allows read access to minidisk 19E of user 'MAINT'          *
*                                                                     *
*     LINK MAINT 19D 19D RR                                           *
*       - Allows read access to minidisk 19d of user 'MAINT'          *
*                                                                     *
*     MDISK 999 3350 236 022 CPPACK WR RPASS WPASS                    *
*       - Defines minidisk with a virtual address of 999              *
*       - Specifies that the minidisk resides on a real 3350          *
*       - Minidisk starts at cylinder 236                             *
*       - Mdisk is 22 cylinders in size (starting at cyl 236          *
*       - The real volume serial number is 'CPPACK'                   *
*       - Mdisk is to be accessed in write mode if no other           *
*         user has write access. Alternate access read-only           *
*       - RPASS is the required password for another user to          *
*         link to this minidisk in read mode                          *
*       - WPASS is the required password for another user to          *
*         link to this minidisk in write mode                         *
***********************************************************************
*
*
*
DIRECTORY 123 3380 XASRES
*
***********************************************************************
*         SYSTEM RESERVED AREAS NOT FOR MINIDISKS                     *
***********************************************************************
*
USER $ALLOC$  NOLOG
 MDISK A01 3380 000 001 XASRES R
 MDISK B01 3380 000 001 XASERV R
 MDISK C01 3380 000 001 XAP001 R
*
USER $DIRECT$ NOLOG
```

```
 MDISK A04 3380 001 002 XASRES  R
*
USER $CP-NUC$ NOLOG
 MDISK A09 3380 003 010 XASRES  R
*
USER $SYSCKP$ NOLOG
 MDISK A06 3380 013 002 XASRES  R
*
USER $SYSWRM$ NOLOG
 MDISK A07 3380 015 002 XASRES  R
*
USER $PAGE$   NOLOG
 MDISK A03 3380 017 024 XASRES R
*
USER $SPOOL$  NOLOG
 MDISK B01 3380 041 087 XASRES R
*
USER $T-DISK$ NOLOG
 MDISK B01 3380 539 020 XASRES R
*
*****************************************************************
*             CMS USERIDS                                       *
*****************************************************************
*
USER CMS1  NOLOG  3M   32M G
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT4 CMSTST
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
*********************************************************************
*                  PROP LOGICAL OPERATOR                            *
*********************************************************************
*
USER LGLOPR  NOLOG 512K 16M ABCDEG
 ACCOUNT ACT1 PROP
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3380 493 001 XASRES WR
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK OPERATOR 191 291 RR
*
*****************************************************************
*             SYSTEM RELATED USERIDS                            *
*****************************************************************
*
USER MAINT NOLOG 16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
 IPL 190
```

```
NAMESAVE GCS
NAMESAVE VTAM
NAMESAVE HELP
NAMESAVE INSTHELP
CONSOLE 009 3215 T
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH  A
SPOOL 00E 1403 A
MDISK 123 3380 0000 2655 XASRES  MW
MDISK 124 3380 0000 2655 XAP001  RR
MDISK 125 3380 0000 2655 XASERV  RR

MDISK 191 3380 0703 016 XASRES  MW
MDISK 193 3380 0176 046 XASRES  MW
MDISK 194 3380 0406 056 XASRES  MW
MDISK 201 3380 0789 022 XASRES  MW RMAINT   WMAINT   MMAINT

MDISK 192 3380 0811 015 XASRES  MW READ WRITE MULTIPLE
MDISK 392 3380 0826 015 XASRES  MW READ WRITE MULTIPLE
MDISK 492 3380 0841 010 XASRES  MW READ WRITE MULTIPLE
MDISK 692 3380 0851 010 XASRES  MW READ WRITE MULTIPLE

MDISK 593 3380 0861 040 XASRES  MW READ WRITE MULTIPLE
MDISK 594 3380 0901 060 XASRES  MW READ WRITE MULTIPLE
MDISK 490 3380 0961 072 XASRES  MW
MDISK 423 3380 0166 010 XASRES  MW
MDISK 293 3380 1033 060 XASRES  MW RCMSAUX  WCMSAUX  MCMSAUX
MDISK 294 3380 1093 080 XASRES  MW RCPAUX   WCPAUX   MCPAUX
MDISK 393 3380 1829 056 XASRES  WR
MDISK 394 3380 1173 130 XASRES  WR
MDISK 19C 3380 1352 029 XASRES  WR
MDISK 49C 3380 1381 029 XASRES  WR
MDISK 596 3380 1410 020 XASRES  WR
MDISK 59E 3380 1430 010 XASRES  WR

MDISK 295 3380 1440 030 XASRES  WR READ WRITE MULTIPLE
MDISK 395 3380 1470 020 XASRES  WR READ WRITE MULTIPLE
MDISK 495 3380 1490 005 XASRES  WR READ WRITE MULTIPLE
MDISK 592 3380 1495 010 XASRES  WR READ WRITE MULTIPLE
MDISK 892 3380 1505 005 XASRES  WR READ WRITE MULTIPLE
MDISK 491 3380 1510 010 XASRES  WR READ WRITE MULTIPLE
MDISK 791 3380 1520 010 XASRES  WR READ WRITE MULTIPLE
MDISK 89E 3380 1530 010 XASRES  WR READ WRITE MULTIPLE
MDISK 896 3380 1540 010 XASRES  WR READ WRITE MULTIPLE
MDISK 895 3380 1550 010 XASRES  WR READ WRITE MULTIPLE

MDISK 391 3380 0306 040 XASRES  MW
MDISK 591 3380 1597 060 XASRES  MW READ WRITE MULTIPLE
MDISK 691 3380 1657 040 XASRES  MW READ WRITE MULTIPLE
MDISK 49D 3380 1697 029 XASRES  MW READ WRITE MULTIPLE
MDISK 501 3380 1726 002 XASRES  MW READ WRITE MULTIPLE
MDISK 291 3380 0346 060 XASRES  MW
MDISK 595 3380 1560 012 XASRES  MW READ WRITE MULTIPLE

MDISK 190 3380 0234 072 XASRES  MW ALL
MDISK 19E 3380 1756 044 XASRES  MW ALL
MDISK 19D 3380 1800 029 XASRES  MW ALL
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
MDISK 36E 3380 0537 002 XASRES RR RPVM     WPVM     MPVM
```

```
 LINK RSCS  191 499  MW
*
USER CMSBATCH NOLOG 1M 2M G
 ACCOUNT 3 SYSTEM
 OPTION ACCT
 IPL CMS PARM AUTOCR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 195 3380 494 002 XASRES MR RBATCH   WBATCH   MBATCH
*
USER GCS NOLOG 16M 16M ABCDEFG
  AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT GCS RECVM
  IPL GCS
  NAMESAVE GCS
  CONSOLE 009 3215
  SPOOL 00C 2540 READER *
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 19D 19D RR
  LINK MAINT 595 595
  LINK MAINT 59E 59E
  MDISK 191 3380 463 004 XASRES MR RGCS WGCS MGCS
*
USER SYSMAINT NOLOG    3M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 SYSPROG
  IPL CMS
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 123 123  MW
 LINK MAINT 191 192  RR
 LINK MAINT 190 190  RR
 LINK MAINT 19E 19E  RR
 LINK MAINT 19D 19D  RR
*
USER OPERATOR NOLOG   16M 32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 2 OPERATOR
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 719 005 XASRES MR
 LINK OP1   191 192 RR
*
*OP1 IS AN ALTERNATE OPERATOR USERID
*
USER OP1 NOLOG 3M  32M ABCDEFG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 3 OPERATOR
```

```
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 724 001 XASRES MR
 LINK OPERATOR 191 192 RR
*
* This userid is for the IBM CE's use in running
* OLTSEP.
* OLTSEP is automatically IPLed in the virtual machine.
* A minimum machine size of one megabyte is required to
* run OLTSEP.
* The privilege class of F allows the CE to specify
* intensive recording mode.
* The console address of 01F is required by OLTSEP.
* The unit record addresses are those required by
* OLTSEP.
* The 5FF minidisk is the CE's OLTSEP pack.
*
USER OLTSEP NOLOG 3M 8M FG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT OLTSEP IBMCE
  IPL 5FF
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  MDISK 5FF 3380 000 2655 CEPACK MR
*
*        **  EREP **
*
* This userid is for the IBM CE's use in running
* CPEREP.
* CMS is automatically IPLed in the virtual machine.
* The 190 minidisk is the CMS system disk.
* The 191 minidisk may be used to save often-used EREP
* control statements and procedures (EXECs).
* THE 190 MINIDISK HAS THE CMS TXTLIBS NECESSARY TO RUN
* EREP.
*
USER EREP NOLOG 3M 8M BFG
 AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT EREP IBMCE
   IPL CMS
  CONSOLE 01F 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH B
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 201 201 RR
  MDISK 191 3380 745 002 XASRES WR READ     WRITE   MULTIPLE
*
USER OPERATNS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 1 OPERATNS
 IPL CMS
```

```
 CONSOLE 009 3215
 SPOOL 00C 2540 READER D
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 725 017 XASRES MR RDVF     WDVF     MDVF
*
*
USER IPCS NOLOG 3M 8M BCEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 8 CE-ROOM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK OPERATNS 193 192 RR
*
USER AUTOLOG1  NOLOG   3M 8M ABCDEG
 AUTOLOG OP1 MAINT
 ACCOUNT 9 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 MDISK 191 3380 742 001 XASRES MR RAUTOLOG WAUTOLOG MAUTOLOG
*
USER DISKACNT NOLOG 3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 743 001 XASRES MR READ WRITE MULTIPLE
*
USER IBMPSR NOLOG  3M 8M BG
 ACCOUNT 10 ACCNTNG
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 744 001 XASRES WR
*
USER IVPM2 NOLOG  3M   4M G
```

```
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER VMUTIL NOLOG  3M 8M BDEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 11 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
*
USER SYSDUMP1 NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 12 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 123 3380 000 2655 XASRES RR
 MDISK 124 3380 000 2655 XAP001 RR
 MDISK 125 3380 000 2655 XASERV RR
*
*
USER RSCS     NOLOG   3M 8M BEG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 13 SYSTEM
  NOPDATA
  IPL 191
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 MDISK 191 3380 535 002 XASRES MR RRSCS    WRSCS    MRSCS
*
USER PVM      NOLOG   3M 8M BG
 AUTOLOG AUTOLOG1 OP1 MAINT
 ACCOUNT 14 SYSTEM
  IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
```

```
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19E 19E RR
 LINK MAINT 19D 19D RR
 LINK MAINT 36E 191 MR
*
*
*****************************************************************
*             OTHER OPERATING SYSTEM USERIDS                    *
*****************************************************************
*
```

Sample XAMAINT Directory Entry for 3380-K

If you are using the procedure in Chapter 4, "Installing VM/XA System Product Release 2 Using an Existing VM/SP or VM/SP HPO System" on page 219 to install your system, add the following entry to your directory:

```
*
USER XAMAINT NOLOG 16M 16M ABCDEFG
 OPTION ECMODE DIAG98
 ACCOUNT 1 SYSPROG
 IPL 190
*NAMESAVE GCS
*NAMESAVE VTAM
*NAMESAVE HELP
*NAMESAVE INSTHELP
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 MDISK 123 3380 0000 2655 XASRES  MW
 MDISK 124 3380 0000 2655 XAP001  RR
 MDISK 125 3380 0000 2655 XASERV  RR

 MDISK 191 3380 0703 016 XASRES  MW
 MDISK 193 3380 0176 046 XASRES  MW
 MDISK 194 3380 0406 056 XASRES  MW
 MDISK 201 3380 0789 022 XASRES  MW RMAINT   WMAINT   MMAINT

 MDISK 192 3380 0811 015 XASRES  MW READ WRITE MULTIPLE
 MDISK 392 3380 0826 015 XASRES  MW READ WRITE MULTIPLE
 MDISK 492 3380 0841 010 XASRES  MW READ WRITE MULTIPLE
 MDISK 692 3380 0851 010 XASRES  MW READ WRITE MULTIPLE

 MDISK 593 3380 0861 040 XASRES  MW READ WRITE MULTIPLE
 MDISK 594 3380 0901 060 XASRES  MW READ WRITE MULTIPLE
 MDISK 490 3380 0961 072 XASRES  MW
 MDISK 423 3380 0166 010 XASRES  MW
 MDISK 293 3380 1033 060 XASRES  MW RCMSAUX  WCMSAUX  MCMSAUX
 MDISK 294 3380 1093 080 XASRES  MW RCPAUX   WCPAUX   MCPAUX
 MDISK 393 3380 1829 056 XASRES  WR
 MDISK 394 3380 1173 130 XASRES  WR
 MDISK 19C 3380 1352 029 XASRES  WR
 MDISK 49C 3380 1381 029 XASRES  WR
 MDISK 596 3380 1410 020 XASRES  WR
 MDISK 59E 3380 1430 010 XASRES  WR

 MDISK 295 3380 1440 030 XASRES  WR READ WRITE MULTIPLE
 MDISK 395 3380 1470 020 XASRES  WR READ WRITE MULTIPLE
 MDISK 495 3380 1490 005 XASRES  WR READ WRITE MULTIPLE
 MDISK 592 3380 1495 010 XASRES  WR READ WRITE MULTIPLE
 MDISK 892 3380 1505 005 XASRES  WR READ WRITE MULTIPLE
 MDISK 491 3380 1510 010 XASRES  WR READ WRITE MULTIPLE
 MDISK 791 3380 1520 010 XASRES  WR READ WRITE MULTIPLE
 MDISK 89E 3380 1530 010 XASRES  WR READ WRITE MULTIPLE
 MDISK 896 3380 1540 010 XASRES  WR READ WRITE MULTIPLE
 MDISK 895 3380 1550 010 XASRES  WR READ WRITE MULTIPLE

 MDISK 391 3380 0306 040 XASRES  MW
 MDISK 591 3380 1597 060 XASRES  MW READ WRITE MULTIPLE
 MDISK 691 3380 1657 040 XASRES  MW READ WRITE MULTIPLE
```

```
 MDISK 49D 3380 1697 029 XASRES  MW READ WRITE MULTIPLE
 MDISK 501 3380 1726 002 XASRES  MW READ WRITE MULTIPLE
 MDISK 291 3380 0346 060 XASRES  MW
 MDISK 595 3380 1560 012 XASRES  MW READ WRITE MULTIPLE

 MDISK 190 3380 0234 072 XASRES  MW ALL
 MDISK 19E 3380 1756 044 XASRES  MW ALL
 MDISK 19D 3380 1800 029 XASRES  MW ALL
*
* MDISK STATEMENTS FOR INSTALL & SERVICE OF RSCS & PASS-THROUGH
 MDISK 36E 3380 0537 002 XASRES RR RPVM     WPVM     MPVM
 LINK RSCS  191 499  MW
 LINK MAINT 190 390  RR
*
```

System Residence DASD Allocation for 3380-K

Figure 62 shows the allocation for the system residence device for the 3380-K VM/XA SP Starter System.

```
TYPE  CYL  CYL
PERM 0000-0000
DRCT 0001-0002
PERM 0003-0013
PAGE 0014-0037
SPOL 0038-0074
PERM 0075-0197
TDSK 0198-0216
PERM 0217-2654
```

Figure 62. System Residence DASD Allocation for 3380-K

Minidisk Maps for 3380-K System Residence Device

Figure 63 shows the minidisk maps of the system residence device for the 3380-K. To map your own system residence device for comparison, issue DISKMAP *fn* DIRECT, where *fn* is the name of your directory.

```
------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
CEPACK   OLTSEP      5FF     3380     0000     2654     2655

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XAP001   $ALLOC$     C01     3380     0000     0000     0001
         MAINT       124     3380     0000     2654     2655 **OVERLAP**
         SYSDUMP1    124     3380     0000     2654     2655 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XASERV   $ALLOC$     B01     3380     0000     0000     0001
         MAINT       125     3380     0000     2654     2655 **OVERLAP**
         SYSDUMP1    125     3380     0000     2654     2655 **OVERLAP**

------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
XASRES   $ALLOC$     A01     3380     0000     0000     0001
         $DIRECT$    A04     3380     0001     0002     0002
         $CP-NUC$    A09     3380     0003     0012     0010
         $SYSCKP$    A06     3380     0013     0014     0002
         $SYSWRM$    A07     3380     0015     0016     0002
         $PAGE$      A03     3380     0017     0040     0024
         $SPOOL$     B01     3380     0041     0127     0087
                                       128      165       38    GAP
         MAINT       423     3380     0166     0175     0010
         MAINT       193     3380     0176     0221     0046
                                       222      233       12    GAP
         MAINT       190     3380     0234     0305     0072
         MAINT       391     3380     0306     0345     0040
         MAINT       291     3380     0346     0405     0060
         MAINT       194     3380     0406     0461     0056
                                       462      462        1    GAP
```

Figure 63 (Part 1 of 3). Minidisk Maps of the System Residence Device for 3380-K

GCS	191	3380	0463	0466	0004	
			467	492	26	GAP
LGLOPR	191	3380	0493	0493	0001	
CMSBATCH	195	3380	0494	0495	0002	
			496	534	39	GAP
RSCS	191	3380	0535	0536	0002	
MAINT	36E	3380	0537	0538	0002	
$T-DISK$	B01	3380	0539	0558	0020	
			559	702	144	GAP
MAINT	191	3380	0703	0718	0016	
OPERATOR	191	3380	0719	0723	0005	
OP1	191	3380	0724	0724	0001	
OPERATNS	191	3380	0725	0741	0017	
AUTOLOG1	191	3380	0742	0742	0001	
DISKACNT	191	3380	0743	0743	0001	
IBMPSR	191	3380	0744	0744	0001	
EREP	191	3380	0745	0746	0002	
			747	788	42	GAP
MAINT	201	3380	0789	0810	0022	
MAINT	192	3380	0811	0825	0015	
MAINT	392	3380	0826	0840	0015	
MAINT	492	3380	0841	0850	0010	
MAINT	692	3380	0851	0860	0010	
MAINT	593	3380	0861	0900	0040	
MAINT	594	3380	0901	0960	0060	
MAINT	490	3380	0961	1032	0072	
MAINT	293	3380	1033	1092	0060	
MAINT	294	3380	1093	1172	0080	
MAINT	394	3380	1173	1302	0130	
			1303	1351	49	GAP

Figure 63 (Part 2 of 3). Minidisk Maps of the System Residence Device for 3380-K

```
MAINT     19C     3380     1352     1380     0029
MAINT     49C     3380     1381     1409     0029
MAINT     596     3380     1410     1429     0020
MAINT     59E     3380     1430     1439     0010
MAINT     295     3380     1440     1469     0030
MAINT     395     3380     1470     1489     0020
MAINT     495     3380     1490     1494     0005
MAINT     592     3380     1495     1504     0010
MAINT     892     3380     1505     1509     0005
MAINT     491     3380     1510     1519     0010
MAINT     791     3380     1520     1529     0010
MAINT     89E     3380     1530     1539     0010
MAINT     896     3380     1540     1549     0010
MAINT     895     3380     1550     1559     0010
MAINT     595     3380     1560     1571     0012
                           1572     1596       25    GAP
MAINT     591     3380     1597     1656     0060
MAINT     691     3380     1657     1696     0040
MAINT     49D     3380     1697     1725     0029
MAINT     501     3380     1726     1727     0002
                           1728     1755       28    GAP
MAINT     19E     3380     1756     1799     0044
MAINT     19D     3380     1800     1828     0029
MAINT     393     3380     1829     1884     0056
MAINT     123     3380     0000     2654     2655 **OVERLAP**
SYSDUMP1  123     3380     0000     2654     2655 **OVERLAP**
```

Figure 63 (Part 3 of 3). Minidisk Maps of the System Residence Device for 3380-K

Sample SPLOAD PROFILE

Information about the product tape layout, fields for userid, minidisk address, format, volume number and tape number are contained in this profile. The following is an example of the SPLOAD profile for the 3420 (6250 BPI) tape.

```
!-------------------------------------------------------------!
!             6250 BPI VM/XA Merged Product Tape              !
!-------------------------------------------------------------!
!
! Volume 1
!
Memo      Files      MAINT     191   XASP20V1   1        2
Install   Tools      MAINT     191   XASP20V1   1        3
Sysgen    Tools      MAINT     193   XASP20V1   1        4
System    Samples    MAINT     191   XASP20V1   1        5
CPLOC     Samples    MAINT     295   XASP20V1   1        6
CMSLOC    Samples    MAINT     395   XASP20V1   1        7
CP        Object     MAINT     194   XASP20V1   1        8
CP        Putapply   MAINT     192   XASP20V1   1        9
CP        Putdelta   MAINT     294   XASP20V1   1       10
CP        Corapply   MAINT     291   XASP20V1   1       11
CP        Cordelta   MAINT     291   XASP20V1   1       12
DUMPVIEW  Object     MAINT     193   XASP20V1   1       13
DUMPVIEW  Putapply   MAINT     392   XASP20V1   1       14
DUMPVIEW  Putdelta   MAINT     293   XASP20V1   1       15
DUMPVIEW  Corapply   MAINT     391   XASP20V1   1       16
DUMPVIEW  Cordelta   MAINT     391   XASP20V1   1       17
!
! Volume 2
!
CMS       Base       MAINT     193   XASP20V2   2        2
CMS       Putapply   MAINT     392   XASP20V2   2        3
CMS       Putdelta   MAINT     293   XASP20V2   2        4
CMS       Corapply   MAINT     391   XASP20V2   2        5
CMS       Cordelta   MAINT     391   XASP20V2   2        6
IOCP      Files      MAINT     193   XASP20V2   2        7
CMS       System     MAINT     190   XASP20V2   2        8
GCS       Object     MAINT     595   XASP20V2   2        9
GCS       Interface  MAINT     193   XASP20V2   2       10
GCS       Putapply   MAINT     592   XASP20V2   2       11
GCS       Putdelta   MAINT     596   XASP20V2   2       12
GCS       Corapply   MAINT     491   XASP20V2   2       13
GCS       Cordelta   MAINT     491   XASP20V2   2       14
AMENGHLP  Files      MAINT     19D   XASP20V2   2       15
UCENGHLP  Files      MAINT     19C   XASP20V2   2       15
!
! Volume 3
!
CP        Source     MAINT     394   XASP20V3   3        2
CMS       Source     MAINT     393   XASP20V3   3        3
DUMPVIEW  Source     MAINT     393   XASP20V3   3        4
!
!
/*-------------------------------------------------------------*/
/*          6250 BPI VM/XA NLS Feature Tape                    */
/*-------------------------------------------------------------*/
/*                                                             */
CMS       Base       MAINT     193   R2M0NLS    1        2
```

```
CP       Object     MAINT     194   R2M0NLS     1          3
HELP     Files      MAINT     ?     R2M0NLS     1          4
CMS      Source     MAINT     393   R2M0NLS     1          5
DUMMY    Object     MAINT     XXX   R2M0NLS     1          6
GCS      Object     MAINT     595   R2M0NLS     1          7
/*                                                          */
/*                                                          */
```

Appendix D. Listing CP Data Areas and Control Blocks

You can get a listing of CP data areas and control blocks written in ASSEMBLER by assembling a file called HCPBLOKS ASSEMBLE. (Data areas and control blocks in PLAS are not listed). Follow these steps:

1. Log onto the MAINT user ID.
2. Define a temporary disk. Use the chart below to determine how much DASD space you need:

Device (devtype)	Cylinders (cyls)
3330	22
3350	10
3375	16
3380	10

```
def devtype vdevno cyls ■
```

Substitute a device type for *devtype*, a virtual device number for *vdevno*, and a cylinder count for *cyls*.

```
DASD vdevno DEFINED
Ready; T=n.nn/n.nn hh:mm:ss
```

3. Format the temporary minidisk and access it as A.

```
format vdevno a ■
```

Substitute the virtual device number for *vdevno*.

```
FORMAT WILL ERASE ALL FILES
 ON DISK 'A(vdevno)'. DO YOU WISH
 TO CONTINUE? (YES|NO):
yes ■
ENTER DISK LABEL:
```

```
label ■
FORMATTING DISK 'A'.
nn CYLINDERS FORMATTED ON 'A(vdevno)'.
Ready; T=n.nn/n.nn hh:mm:ss
```

Enter your own disk label for this temporary disk.

4. Access MAINT's 394 minidisk (the CP source file disk) and copy the HCPBLOKS ASSEMBLE file.

```
access 394 c ■
Ready; T=n.nn/n.nn hh:mm:ss
```

```
copy hcpbloks assemble c (olddate unpack ■
Ready; T=n.nn/n.nn hh:mm:ss
```

You must access 394 as C. The file you copy is in packed format. Use the UNPACK option to unpack the file.

5. Access MAINT's 194 minidisk (the control file and MACLIB disk) and 294 minidisk (the update file disk).

```
access 194 b/a ■
B (194) R/O
Ready; T=n.nn/n.nn hh:mm:ss
access 294 c/a ■
'294' REPLACES ' C (394) '
C (294) R/O
Ready; T=n.nn/n.nn hh:mm:ss
```

6. Reaccess MAINT's 394 minidisk as a read-only extension to MAINT's A minidisk.

```
access 394 d/a ■
D (394) R/O
Ready; T=n.nn/n.nn hh:mm:ss
```

7. Check MAINT's minidisk access order. The minidisks, access modes, and link mode (R/W or R/O) should be the same as shown.

```
query search ■

label   vdevno  A    R/W
MNT194  194     B/A  R/O
MNT294  294     C/A  R/O
MNT394  394     D/A  R/O
MNT190  190     S    R/O
MNT19E  19E     Y/S  R/O
Ready; T=n.nn/n.nn hh:mm:ss
```

label and *vdevno* are the values you substituted when you defined and formatted the temporary minidisk.

8. Use the UPDATE command to update HCPBLOKS ASSEMBLE:

```
update hcpbloks assemble * hcpxa cntrl * (ctl ■
UPDATING 'HCPBLOKS ASSEMBLE D1'
APPLYING 'HCPBLOKS ft       C1'
Ready; T=n.nn/n.nn hh:mm:ss
```

This example assumes there is an update file for HCPBLOKS. If there is an update file, it will have a special file type (*ft*). If there is no update file, you will receive the message NO UPDATE FILES WERE FOUND.

9. Issue the GLOBAL command for the macro libraries called by HCPBLOKS:

```
global maclib hcpxa dmssp cmslib osmacro osmacro1 ■
Ready; T=n.nn/n.nn hh:mm:ss
```

10. Set the virtual punch to stay open after spool files reach 50,000 records:

```
spool punch noeof ■
```

The option NOEOF means the punch stays open after punch spool files reach 50,000 records.

11. Issue the HASM command for HCPBLOKS:

```
hasm fn (sysparm (exp) xref (full) print ■
```

If updates were applied to HCPBLOKS ASSEMBLE, substitute $HCPBLOK for *fn*. If updates were not applied to HCPBLOKS ASSEMBLE, substitute HCPBLOKS for *fn*.

```
PRT FILE spoolid SENT FROM MAINT PRT AS spoolid RECS nnnn COPY  001 A NOHOLD NOKEEP
Ready; T=n.nn/n.nn hh:mm:ss
```

Appendix E. Example of Alternate GCS Nucleus Placement

Overview

This appendix demonstrates how to save your GCS system at a virtual storage location other than the one provided by the product tape samples. You might want the GCS segment at a different location, depending on the other segments being used by your GCS group.

This section also describes how to increase the size of the GCS saved system. One reason that you might want to do this is to increase the amount of available common storage. Refer to *VM/XA SP Planning and Administration* for information about calculating common storage requirements for GCS.

To relocate your GCS saved system and change its size, you need to:

- Change the SLC names in the GCSLOAD EXEC
- Create SLC files to contain the new address locations
- Modify the DEFSYS entries in the SAMPNSS EXEC for GCS
- Build and save the GCS nucleus.

Procedure

This sample procedure demonstrates how to relocate your GCS saved system in a 16-megabyte virtual machine. Originally 2M in size and loaded at X'400000', the segment is moved to X'800000' and increased in size to 3MB.

1. Log on as MAINT and access the GCS system disk:

   ```
   access 595 a ■
   ```

2. Modify the GCS loadlist (GCSLOAD EXEC A) and change the Set Location Counter (SLC) values that determine where the shared portion of GCS is loaded into virtual storage.

   In the commands that follow, "SLC L400000" is the name of a CMS file that contains the address of the starting location for loading the main portion of the GCS nucleus. This SLC statement precedes the CSIALP entry in the loadlist. "SLC L600000" marks the end of the GCS nucleus (also the end of common storage), and it precedes the CSIZET entry.

   Note: The GCS loadlist may be changed by service. After applying a PUT, you should check the file and change it if necessary before you rebuild the GCS nucleus.

   Enter the following commands:

   ```
   xedit gcsload EXEC a ■
   set case upper ■
   top ■
   change / SLC L400000 / SLC L800000 /* ■
   top ■
   change / SLC L600000 / SLC LB00000 /* ■
   file ■
   ```

3. Create two new SLC files to match the new loadlist:

   ```
   xedit SLC L800000 a ■
   ```

   ```
   input $SLC  800000 ■
   set hex on ■
   ```

   There must be **two** blanks between SLC and the address.

   ```
   change /$/X'02'/ ■
   file ■
   xedit SLC LB00000 a ■
   ```

   X'02' is an unprintable loader control character.

   ```
   input $SLC  B00000 ■
   ```

   There must be **two** blanks between SLC and the address.

   ```
   set hex on ■
   change /$/X'02'/ ■
   file ■
   ```

   X'02' is an unprintable loader control character.

4. Modify the DEFSYS entry for your GCS saved system in the SAMPNSS EXEC. As a result, you may also have to change other DEFSYS entries. In this example, the new location for GCS overlays the default locations for CMSVSAM and CMSAMS, so you will have to change those entries.

   Note: To convert an entry in the system name table of a VM/SP or VM/SP HPO system to a DEFSYS entry, see *VM/XA SP Conversion Notebook*.

   The IBM-supplied DEFSYS entry for GCS in the SAMPNSS EXEC looks like this:

   ```
   'CP DEFSYS GCS  0-6 EW 400-5FF SW MINSIZE=256K VMGROUP RSTD'
   ```

   GCS occupies pages 400 through 5FF (all of segments 4 and 5).

   a. Calculate the new values for the DEFSYS entry:

      1) To calculate the page number where the GCS nucleus begins, divide the corresponding SLC value (X'800000' in this example) by the page size, X'1000'. The result is X'800'.

      2) To calculate the page number where GCS ends, divide the corresponding SLC value (X'B00000' in this example) by X'1000' and subtract one. The result is X'AFF'.

   b. XEDIT the SAMPNSS EXEC and enter the new values in the DEFSYS entry for GCS. The changed entry looks like this:

      ```
      'CP DEFSYS GCS  0-6 EW 800-AFF SW MINSIZE=256K VMGROUP RSTD'
      ```

5. Copy the latest version of the product parameter file to the 191 minidisk:

   copy 56643082 $ppf *fm* **= = a (replace** ■

6. Generate and save the GCS nucleus:

   access 495 c ■ — You do not want to change the nucleus on the 595 disk.

   Note:

7. Before you generate the nucleus, you may want to use GROUP EXEC to create a new GCS configuration file for this nucleus. See the GCS step in the installation procedure.

   itask build gcs *systemname* ■ — If you do not specify a *systemname*, the default is GCS.
   ipl cms ■

Appendix F. Restricted Logon Passwords

The CP nucleus includes a system security feature called ADRP (Auto-Deactivation of Restricted Passwords). ADRP works with a CMS file named RPWLIST DATA that contains the list of restricted logon passwords shown in Figure 64. You can edit this file to add your own restricted passwords.

Issuing the DIRECTXA command causes the system to search the directory for the restricted passwords contained in this list. All passwords that match are changed to NOLOG in the directory before the directory is placed online. You cannot log on to any user ID whose password has been changed to NOLOG. Therefore, make sure that you have changed **all** restricted passwords in the directory to unique non-restricted passwords before you issue the DIRECTXA command.

```
ACNT     ************************************************************
AUTOLOG  * THIS MODULE IS RESTRICTED MATERIALS OF IBM.               *
AUTOLOG1 * 5664-308 (C) COPYRIGHT IBM CORPORATION - 1988            *
BATCH    * LICENSED MATERIALS - PROPERTY OF IBM                     *
CE       * SEE COPYRIGHT INSTRUCTIONS, G120-2083                    *
CMSUSER  ************************************************************
CMS1
CMS2     ************************************************************
CMS3     *                                                          *
CPCMS    *                Restricted Password List                  *
DIRM     *  Format Rules:                                           *
ECMODE   *                                                          *
GCS      *   1) The RPWLIST DATA file must be fixed record length   *
IBMCE    *      format, with a record length of at least 8.         *
IPCS     *   2) Each password must start in column 1.               *
ISMAINT  *   3) Columns 1-8 must contain restricted passwords only. *
ITPS     *   4) Each line may contain only one password.            *
IVPASS   *   5) Column 9 must contain a blank.                      *
LEV2VM   *   6) Columns 10 through 80 may be used for comments, such *
MAINT    *      as this prologue.                                   *
MASTER   *                                                          *
MDVR     ************************************************************
OPASS
OPERATNS
OPERATOR
OSVS1
PASSWORD
PRODBM
PROMAIL
PSR
ROUTER
RSCS
SFBATCH
SSFCAL
SQLDBAPW
SQLUSER
SYSADMIN
SYSDUMP
VMAP
VSEIP
VSEIPO
VSEMAINT
```

Figure 64. RPWLIST DATA File

Appendix G. Controlling Disk String Merges

Automatic Merging

VMFREC normally merges the alternate DELTA, APPLY, and LOCAL minidisks to their corresponding current disks, in accordance with the :MERGE. tag in the product parameter file.

Since disks are merged only within a string (for example, DELTA disks are merged to DELTA disks, not to APPLY disks), merging does not consolidate the contents of all disks, but saves back levels of service on the current disks and frees the alternate disks for new levels of service. Merging also frees the alternate DELTA disk, because VMFREC loads a new level of service files there.

Note: When you apply corrective service, the DELTA and APPLY disk strings are redefined to consist of the LOCAL1 disks in the temporary product parameter file. This redefinition is necessary in order for VMFREC to load files to the LOCAL1 disk. It is done automatically by the corrective service overrides in the product parameter file.

Preventing Automatic Merging

You may not want VMFREC to merge a certain disk string. A common reason is that you apply local service on a different schedule from program update service and corrective service, and you want to keep your LOCAL1 disks unchanged.

To prevent VMFREC from merging a string, define and format a minidisk at any address you are not using. Don't put anything on this minidisk. Then put this disk at the head of the string you don't want to merge in the product parameter file. (Remember that if you are processing corrective service, you must make the change in the corrective override section of the product parameter file, not in the base.) When VMFREC finds an empty disk at the head of the string, it will not do the merge.

If you do not want VMFREC to merge any disk strings at all, remove all the symbolic disk string names from the :MERGE. tag in the product parameter file. For example, change:

```
:MERGE. DELTA1 APPLY LOCAL1
```

to:

```
:MERGE
```

Manual Merge Procedure

For Program Update Service

For the addresses of the DELTA1, APPLY, and LOCAL1 minidisks merged in this procedure, see Figure 6 on page 364.

1. Copy the contents of the alternate DELTA1 disk to the current DELTA1 disk, using the REPLACE option.
2. Erase all the contents of the alternate DELTA1 disk. When VMFREC finds this disk empty, it will not merge the DELTA1 string.
3. Copy the contents of the alternate APPLY disk to the current APPLY disk, using the REPLACE option.

4. Erase all the contents of the alternate APPLY disk. When VMFREC finds this disk empty, it will not merge the APPLY string.
5. Copy the contents of the intermediate alternate LOCAL1 disk to the current LOCAL1 disk, using the REPLACE option.
6. Erase all the contents of the intermediate alternate LOCAL1 disk.
7. Copy the contents of the alternate LOCAL1 disk to the intermediate alternate LOCAL1 disk, using the REPLACE option.
8. Erase all the contents of the alternate LOCAL1 disk. When VMFREC finds this disk empty, it will not merge the LOCAL1 string.

For Corrective Service

For the addresses of the LOCAL1 minidisks merged in this procedure, see Figure 6 on page 364.

1. Copy the contents of the intermediate alternate LOCAL1 disk to the current LOCAL1 disk, using the REPLACE option.
2. Erase all the contents of the intermediate alternate LOCAL1 disk.
3. Copy the contents of the alternate LOCAL1 disk to the intermediate alternate LOCAL1 disk, using the REPLACE option.
4. Erase all the contents of the alternate LOCAL1 disk. When VMFREC finds this disk empty, it will not merge the LOCAL1 string (which also serves as the DELTA1 and APPLY strings during corrective service).

Appendix H. Service Reference Tables

VM/XA SP File Types and Abbreviations

The first column of Table 22 shows the 3-character abbreviations conventionally used for the file names in the second column. These abbreviations are used in the file type of parts supplied for service (see "Parts Supplied For Service and What to Do with Them" on page 863).

Table 22 (Page 1 of 2). VM/XA SP File Types and Abbreviations

Abbreviated File Type	Complete File Type
$CN	$CONSTS
$EX	$EXEC
$PF	$PPF
$XE	$XEDIT
ASM	ASSEMBLE
AXA	AUXXA
AXM	AUXMXA
CON	CONSTS
CPY	COPY
CTL	CNTRL
DAT	DATA
DLB	DOSLIB
DLK	DOSLNK
DRC	DIRECT
DSF	DSF
EAM	EXCAMENG
EUC	EXCUCENG
EXC	EXEC
EXL	EXECLIST
HAB	HELPABBR
HCM	HELPCMS
HCP	HELPCP
HCQ	HELPCPQU
HCS	HELPCPSE
HDI	HELPDEFI
HDS	HELPDISP
HDU	HELPDUMP
HED	HELPEDIT

Table 22 (Page 1 of 2). VM/XA SP File Types and Abbreviations

Abbreviated File Type	Complete File Type
HEP	HELP
HEX	HELPEXEC
HE2	HELPEXC2
HGR	HELPGROU
HHE	HELPHELP
HIN	HELPINDI
HLP	HLP
HMA	HELPMACR
HME	HELPMENU
HMO	HELPMONI
HMQ	HELPCMSQ
HMS	HELPMSG
HPF	HELPPF
HPR	HELPPREF
HPU	HELPPURG
HQU	HELPQUER
HRX	HELPREXX
HSE	HELPSET
HSR	HELPSRPI
HSS	HELPCMSS
HST	HELPSTOR
HTR	HELPTRAC
HTS	HELPTASK
HWT	HCPSADWT
HXE	HELPXEDI
LAN	LANGUAGE
LDM	LOADMAP
LDR	LOADER

Table 22 (Page 2 of 2). VM/XA SP File Types and Abbreviations

Abbreviated File Type	Complete File Type
LEF	LEF0000
LKC	LKEDCTRL
LKE	LKEDIT
LOL	LOADLIB
L00	L000000
L03	L00E000
L20	L020000
L23	L023000
L40	L400000
L50	L500000
MAC	MACRO
MAP	MAP
MEM	MEMO
MLB	MACLIB
MOD	MODULE
PRF	PROFILE
PRS	PRODUCTS
REP	REPOS
SAE	SAMPEXEC
SAP	SAMPLIST
SCR	SCRIPT
SCU	SCRIPTUC
SOR	SOURCE
SYN	SYNONYM
TLB	TXTLIB
TXT	TEXT
XAM	XAMAINT
XED	XEDIT

Parts Supplied For Service and What to Do with Them

Table 23 shows the parts that IBM will supply for service to each part of your system. The first column identifies the part to be serviced by file type. The second column shows whether that kind of part is serviced by update, replacement, or both. The third and fourth columns list the parts that IBM will send you when you apply corrective service and program update service respectively. The fifth column shows the parts required for update service: the source file (for example, an ASSEMBLE file), which you already have, the update, which is supplied on the COR or PUT tape, and the auxiliary control file, which is generated during the service process. The sixth column shows the most important EXECs and commands you will use in rebuilding the serviced part. (For full instructions, see Chapter 13, "Rebuilding CMS after Applying Service" on page 457, Chapter 14, "Rebuilding CP after Applying Service" on page 515, Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility" on page 537, or Chapter 16, "Program Update Service or Corrective Service to GCS" on page 547.) If the sixth column is blank, the part supplied by IBM simply replaces the one you already have. If you want to keep your own part for reference, you can rename it.

Table 23 (Page 1 of 2). Parts Supplied For Service and What to Do with Them

Usable Form	Update or Replace?	Parts For COR Service	Parts For PUT Service	Update Parts	Build Function
EXEC	Update $EXEC	update AUXXA EXC*nnnnn* EXEC $EX*nnnnn*[1] $EXEC[1]	update AUXXA EXC*nnnnn* EXEC $EX*nnnnn*[1] $EXEC[1]	update AUX*xxxxx* $EXEC	EXECUPDT Copy *fn* EXC*nnnnn fm* = EXEC =[2]
	Replace EXEC	EXC*nnnnn* EXEC	EXC*nnnnn* EXEC		Copy *fn* EXC*nnnnn fm* = EXEC =
XEDIT	Update $XEDIT	update AUXXA XED*nnnnn* XEDIT $XE*nnnnn*[1] $XEDIT[1]	update AUXXA XED*nnnnn* XEDIT $XE*nnnnn*[1] $XEDIT[1]	update AUX*xxxxx* $XEDIT	EXECUPDT Copy *fn* XED*nnnnn fm* = XEDIT =[2]
	Replace XEDIT	XED*nnnnn* XEDIT	XED*nnnnn* XEDIT		Copy *fn* XED*nnnnn fm* = XEDIT =
MACLIB	Update MACRO	update AUXXA MAC*nnnnn*[1] MACRO[1]	update AUXXA MAC*nnnnn*[1] MACRO[1]	update AUX*xxxxx* MACRO	VMFMAC
	Replace MACLIB		MACLIB		
COPY	Update COPY	update AUXXA CPY*nnnnn*[1] COPY[1]	update AUXXA CPY*nnnnn*[1] COPY[1]	update AUX*xxxxx* COPY	VMFMAC

Table 23 (Page 2 of 2). Parts Supplied For Service and What to Do with Them

Usable Form	Update or Replace?	Parts For COR Service	Parts For PUT Service	Update Parts	Build Function
MODULE	Update ASSEMBLE	update TXT*nnnnn* ASM*nnnnn*[1] ASSEMBLE[1]	update TXT*nnnnn* ASM*nnnnn*[1] ASSEMBLE[1] MODULE[1]	update AUX*xxxxx* ASSEMBLE	CMSGEND UTILITY GENMOD
	Replace text deck	TXT*nnnnn*	TXT*nnnnn* MODULE		Copy *fn* TXT*nnnnn fm* = TEXT =
	Replace MODULE	MOD*nnnnn* MODULE	MOD*nnnnn* MODULE		
NUCLEUS	Update ASSEMBLE	update TXT*nnnnn* ASM*nnnnn*[1] ASSEMBLE[1]	update TXT*nnnnn* ASM*nnnnn*[1] ASSEMBLE[1]	update AUX*xxxxx* ASSEMBLE	VMFBLD HCPLDR
	Replace text	TXT*nnnnn*	TXT*nnnnn*		VMFBLD HCPLDR
TEXT	Update ASSEMBLE	update TXT*nnnnn* ASM*nnnnn*[1] ASSEMBLE[1]	update TXT*nnnnn* ASM*nnnnn*[1] ASSEMBLE[1]	update AUX*xxxxx* ASSEMBLE	VMFHASM Copy *fn* TXT*nnnnn fm* = TEXT =[2]
	Replace text deck	TXT*nnnnn*	TXT*nnnnn*		Copy *fn* TXT*nnnnn fm* = TEXT =
$PPF	Replace $PPF	$PF*nnnnn* $PPF	$PF*nnnnn* $PPF		Copy *fn* $PF*nnnnn fm* = $PPF =
Other parts	Replace part	*aaannnnn* real file type	*aaannnnn* real file type		Copy *fn* *aaannnnn ft* = *ft* =

nnnnn is the number of the most recently applied PTF.

xxxxx is one to five alphanumeric characters.

aaa is the text deck file type prefix from the AUX file identification record in the main control file.

Notes:

[1] This is a base part. This part is shipped from IBM only when it is a new part or when the part (source) is replaced.

[2] This procedure is used when there are no local modifications.

Appendix I. How To Find the PTF Number From the APAR Number

If you know the APAR number associated with an update, you can find the corresponding PTF number. For example, suppose that you want to find the PTF number for APAR 01010.

1. Find the file name of a module affected by the APAR. (If you are looking for the PTF number associated with an update listed in a DEPEND entry, you already know one.)

```
listfile * *01010* * ■
HCPABC   H01010HP fm
⋮
```

2. List all text decks with the file name of that module:

```
listfile hcpabc txt* * (EXEC ■
```

The **EXEC** option saves the list in a file called CMS EXEC. Any previously existing CMS EXEC on your A-disk is erased.

3. Look at each text deck until you find one that lists the APAR you are looking for:

```
xedit hcpabc txtnnnnn ■
locate /01010 ■
```

```
H01010HP  COR  UM00000  * Comments
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
H34567HP  COR  UM00001  * Comments
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
H56789HP  COR  UM00002  * Comments
* PREREQ: NONE
* CO-REQ: NONE
* IF-REQ: NONE
* DEPEND: H01010HP
H88888HP  COR  UM00003  * Comments
⋮
(Executable text is here.)
```

The APAR number is the five numeric characters in the first word of the text deck line. The corresponding PTF number is the third word of the same line.

Appendix J. Messages

This appendix contains messages issued by the service EXECs. For other messages, see the *VM/XA SP System Messages and Codes Reference.*

002E [Input|Overlay] {File[(s)]|Dataset| Note} [*fn* [*ft* [*fm*]]] not found.

Explanation: The specified file was not found on the accessed disk(s). Either the file does not reside on this disk, the file identification was misspelled, or incomplete identification was provided to cause the appropriate disk to be searched, or system disk was not accessed as a read-only extension of the A-disk.

For the PRELOAD command, either the loadlist EXEC, the CNTRL file, or one of the input text files could not be found.

For SETPRT command, the module represented by 'fn ft' does not exist in the current CMS Disk Search Order.

For the STATEW command, the file may exist, but it is not on any of the user's read/write disks.

For the ZAP command, either none of the libraries specified for a TXTLIB or LOADLIB could be found, or the INPUT file name could not be located via the STATE macro.

For the ZAPTEXT and EXPAND commands, the input text file or INPUT file name could not be located via the ESTATE command.

For the VMFLKED command, either you specified a file that cannot be found on a minidisk in the CMS search hierarchy, or you specified a file name on a %CONTROL statement as the name of a CNTRL file and that file was not found.

For the VMFPLC2 command, the STOP option has been specified with the LOAD function, and the file was not found in alphabetic sequence.

For the CONVERT command, the input DLCS file you specified was not found.

See the *VM/XA SP CMS Command Reference* for a description of the file identification required by each command and the search procedure used. For the ASM3705, ZAP, ZAPTEXT, and EXPAND commands, see the *VM/XA SP Installation and Service.*

System Action: RC=28.
Execution of the command is terminated. The system status remains the same.

For DMSSPR, nothing has been sent to the virtual 3800.

For DMSLIO, some loader information fields have been initialized, but they should not interfere with a subsequent LOAD command.

For the CONVERT command, conversion stops.

RC=44.

For the VMFPLC2 command, the tape is positioned immediately before the next file.

For the VMFLKED and VMFZAP commands, processing ends.

For the VMFMERGE command, other required files are checked and then processing ends.

User Response: Find or create the desired file. To make sure that the file exists, issue STATE fn ft * or LISTFILE fn ft *. Correct and reissue the command.

For DMSSPR, access the disk having the required module or respecify a different module in the calling sequence and then reissue the SETPRT command.

For a DMSROS TEXT file, ensure that the file is accessible and reissue the command.

For the VMFLKED command, make certain that the proper disks are accessed and check the name of the specified file. If the name was specified incorrectly, re-issue the command with the correct name.

For VMFTXT:

If the file type is EXEC, make sure that a memberlist EXEC file exists and that the file name of the memberlist and the libname parameter are spelled the same. Correct the error and reissue the command.

If the file type is CNTRL, make sure that the specified CNTRL file exists and is correctly spelled. Correct the error and reissue the command.

If the file name and file type pair is one of the following:

VMFMSGS EXEC
VMFDATE MODULE
VMFTXT DATA

contact your system programmer and arrange to have these files installed again on the CMS system disk as file mode 2 files.

For the VMFZAP, VMFMERGE, and VMFREMOV commands, see if the proper disks are specified in the VMFPARM file and then re-issue the command.

For the CONVERT command, correct the file name or access a disk where the file can be found.

003E Invalid option: *option.*

Explanation: The specified option is invalid. It may have been misspelled, or, if the option is truncatable, it may have been truncated improperly, or it may conflict with another option in the command line.

System Action: RC = 24.

Execution of the command is terminated. The system status remains the same.

For DMSLIO, some option processing may have caused such things to happen as user storage to be cleared or the location counter to be set. This should not interfere with a subsequent LOAD command.

For the VMFLKED command, processing ends.

User Response: Correct and re-enter the command.

050E Parameter missing after *value.*

Explanation: A parameter that is required by the command was not specified.

For the ASSGN command, the disk mode must be specified for the SYSaaa logical unit.

For the DLBL command, the disk mode or DUMMY or CLEAR must be specified after the ddname.

For the FILEDEF command, the device name or DUMMY or CLEAR must be specified after the ddname.

For the NUCXDROP command, a required parameter that must follow a function is missing.

For the SET command, a required parameter that must follow a function is missing.

For the XMITMSG command, one of the options required a value to follow it, but the end of the parameter list was reached.

System Action: RC=24.

Execution of the command is terminated. The system status remains the same.

User Response: Correct and reissue the command.

056E File *fn ft* **contains invalid [name|alias|entry|ESD] record formats.**

Explanation: For DMSLBM and DMSNCP (GEN, ADD, REP), the specified file is not in the expected format. MACRO and MEND cards must be included in the MACRO files, and the prototype card must be specified with a name that does not exceed eight characters. If an &COPY statement appears, it must contain a name. A MACLIB must contain "LIB" in columns 4-6 of record one.

For DMSLBT, the specified file has more than 255 entry points (ESD only), or has records which are incompatible or missing. The NAME field in the CSECT instruction of the specified file must have a valid symbol or label.

For DMSLIO, an invalid condition was found in a TEXT or TEXTLIB file. TXTLIB files created on EDF disks must have "PDS" in columns 4-6 of record one. TXTLIB files created on non-EDF disks must have "LIB" in columns 4-6 of record one. RLD data must be compatible with the TEXT file or TXTLIB member to which is belongs. If an ICS statement was submitted, the specified name was previously defined, or the initial length of the CSECT was not found in the ESD card.

For DMSSYN, the specified file is not in the expected format. The SYNONYM file must contain 80-byte records in free form format, with columns 73-80 ignored. The data consists of a command name followed by a blank and the user synonym. This may optionally be followed by a count which is preceded by at least one blank.

For DMSZAP, either the header record for TXTLIB or LOADLIB was invalid, or the pointer to the directory or module map was in error.

For the VMFTXT command, the memberlist EXEC file was not in the required format.

System Action: RC=32.

Execution of the command is terminated. The system status remains the same. For DMSGLB, the library is not globally searched, and the operation continues for any other libraries named in the command.

For the VMFTXT command, the invalid record is ignored. Processing continues for any remaining records in the file.

User Response: For DMSLBM and DMSNCP, issue the MACLIB COMP command, then check the MACLIB with a MACLIB MAP command. Correct the format error.

For DMSGLB, the specified library does not have "LIB" in columns 1-3 or 4-6 of the first record. One possible cause is the library may be in packed format. Correct the library and reissue the command.

For DMSSYN, correct the format of the file.

For DMSLIO, recreate the TXTLIB or TEXT file.

For DMSLBT, if the message specifies ESD, check for more than 255 entry points for a member; otherwise, check for invalid or missing records. If the NAME field in the CSECT instruction was left blank, enter a valid symbol or label.

For the VMFTXT command, correct the invalid entry in the memberlist EXEC file. If the member specified in the invalid record has a file type of TEXT, you may issue the:

```
TXTLIB VMFTXT ADD membername
<(FILename<)>>
RENAME VMFTXT TXTLIB A libname
TXTLIB A
```

commands. If the file type is not TEXT, then erase VMFTXT TXTLIB A and then reissue the command.

For DMSZAP, recreate the library or module.

Then reissue the command.

065E *option* **option specified twice.**

Explanation: The option was specified more than once in the command line.

System Action: RC=24.

Execution of the command is terminated. The system status remains the same.

User Response: Reissue the command, specifying the option only once.

1801E **There is no tape mounted on 181. Mount the correct tape and restart the receive procedure.**

Explanation: VMFREC expects a tape to be mounted on 181, and there was no tape mounted.

System Action: VMFREC exits with RC = 100.

User Response: Mount the tape to be received and restart VMFREC.

1802R **The tape mounted is at a lower service level than the existing service map.**
The service tape is at service level *level*
The SERVICE DISKMAP file indicates service level *service-level*
Is this what you want? (YES/NO)

Explanation: VMFREC will receive data from a PUT that is at a lower level than was previously received.

System Action: If the response to the prompt is NO, then VMFREC exits. Otherwise, the receive procedure continues.

User Response: Respond to the prompt by entering either YES or NO.

1803E **The tape is in the wrong format. Mount the correct tape and restart the receive procedure.**

Explanation: VMFREC expects the first tape file to contain a file that determines the tape type. This file is either incorrect, or is not present.

System Action: VMFREC exits with RC = 100.

User Response: Mount the correct tape.

1804I **Receiving service for {product** *prodid*|**component** *compname* **of product** *prodid*}.

Explanation: This is an informational message indicating which product service is being received for.

System Action: VMFREC continues receiving service.

User Response: None.

1805E **No|Not enough parameters were entered. Check the syntax and re-enter the command.**

Explanation: An attempt was made to execute a procedure, but the required parameters were not entered.

System Action: The procedure exits with RC = 8.

User Response: Review the documentation and restart the procedure with correct parameters.

1806I **The current SERVICE DISKMAP file contains the map of the mounted tape.**

Explanation: This is an informational message, indicating that the tape currently mounted has been previously mapped.

System Action: VMFREC continues receiving service.

User Response: None.

1807E *routine* **cannot continue; the** *fn ft* **file was not found.**

Explanation: A procedure requires a file that cannot be found on an accessed minidisk.

System Action: The procedure exits with RC = 28.

User Response: The file named in the message needs to be made available. Verify that the minidisk access is as it should be. Check the product parameter file to verify that the correct minidisk is listed in the component minidisk assignment (MDA) section.

1808E **The** *ppfname* **$PPF product parameter file was not found.**

Explanation: A procedure requires a product parameter file in order to set required parameters, and the file cannot be found on an accessed minidisk.

System Action: The procedure exits with RC = 28.

User Response: The file named in the message needs to be made available. Verify that the minidisk accesses are as they should be.

1809E **A** *function* **tape error has occurred with return code** *rc*.

Explanation: VMFREC has encountered a serious tape error.

System Action: The procedure exits with the return code received from the VMFPLC2 command.

User Response: Correct the tape drive problem as indicated by *rc* from the VMFPLC2 command. If drive is fine then the tape might not be in the correct format for the procedure.

1810R **Enter a component name or type QUIT.**

Explanation: The procedure needs a component name in order to carry out its function.

System Action: The procedure waits for a valid component name to be entered or QUIT.

User Response: Enter a component name listed in the product parameter file for the product requested.

1811W **The access of** *vdev* **failed for product** *prodid:*

Explanation: An access command has failed for the minidisk listed in the message text.

System Action: The procedure exits with RC = 4.

User Response: Verify that the minidisk is linked correctly.

1812E **Service for** *prodid* **is not on the mounted tape but the** *prodid* **product ID is in the SERVICE DISKMAP file. It is on the service tape mapped with a relative tape number of** *number*.

Explanation: The wrong tape volume is mounted on tape 181.

System Action: The procedure exits with RC = 28.

User Response: Mount the correct tape volume and restart the procedure.

1813E *option1* **and** *option2* **are conflicting options. Check the syntax and re-enter the command.**

Explanation: Conflicting options have been entered.

System Action: The procedure exits with RC = 8.

User Response: Check the documentation, then restart the procedure using correct command syntax.

1814E **Service for the product ID** *prodid* **is on the mounted tape, but the** *prodid* **product ID is not in the SERVICE DISKMAP file.**

Explanation: The tape has not been mapped correctly.

System Action: VMFREC exits with RC = 28.

User Response: Erase the existing service map and restart VMFREC.

1815E *parameter* **is an unknown parameter. Check the syntax and re-enter the command.**

Explanation: An unknown parameter was detected by the procedure.

System Action: The procedure exits with RC = 8.

User Response: Check the documentation, then restart the procedure using correct command syntax.

1816E *parameter* **is an unknown parameter. This product ID is not on the tape or in the SERVICE DISKMAP file.**

Explanation: The first or only parameter entered is invalid.

System Action: The procedure exits with RC = 8.

User Response: Check that the product ID that was entered is valid.

1817E **The** *updateid-name* **update ID is not in the** *fn* **CNTRL file.**

Explanation: The update ID entered on the command line, or listed in the product parameter file, was not found in the control file listed in the product parameter file.

System Action: The procedure exits with RC = 28.

User Response: Verify that the correct update ID was entered, either on the command line or in the product parameter file, according to the procedure syntax. Check that the correct control file is listed in the component section of the product parameter file.

1818E **The product service header file cannot be found. The tape positioning might be in error. Restart the receive procedure beginning with service for product ID** *prodid.*

Explanation: VMFREC cannot find the product header file on the tape, in the tape file calculated.

System Action: VMFREC exits with RC = 28.

User Response: The tape positioning might be incorrect upon return from an unsupported product service EXEC. If this is thought to be the case, then restart the procedure, starting with the product listed in the message text. If this is not the case, then the file counts in the component list of the product parameter file might be incorrect.

1819E **The control file** *fn ft* **was not found.**

Explanation: The control file listed in the component section of the product parameter file cannot be found.

System Action: The procedure exits with RC = 28.

User Response: Verify that the control file listed is correct, and that the minidisks are accessed correctly.

1820E **The build list *fn ft* was not found.**

Explanation: The build list listed in the component section of the product parameter file cannot be found.

System Action: The procedure exits with RC = 28.

User Response: Verify that the build list listed is correct and that the minidisks are accessed correctly.

1821E **All file modes are being used. The previous access order will be restored.**

Explanation: Access to required minidisks failed because no file modes were available.

System Action: The procedure exits with RC = 100.

User Response: Release unwanted minidisks. Reduce the number of minidisks listed, as required in the component section of the product parameter file.

1822W **Access to *vdev* cannot be restored.**

Explanation: Upon exit, the procedure could not reaccess the minidisk listed.

System Action: The procedure continues with RC = 4.

User Response: Upon exit, reaccess the minidisk if needed.

1823W **The *fn ft* PTF parts list file was not found. A PTF might be missing for product ID *prodid*.**

Explanation: A file listed in a PTF parts list could not be found.

System Action: The procedure continues with RC = 4.

User Response: Investigate why the PTF part is missing. Check the minidisk assignments (MDA) section for the component in the product parameter file.

1824E ***vdev*, accessed as *fm*, is not R/W.**

Explanation: The procedure requires the minidisk listed in the message text to be read/write.

System Action: The procedure exits with RC = 100.

User Response: Reaccess the minidisk as read/write, and restart the procedure.

1825W **A part handler cannot be found for part *part-type* while applying PTF *ptfnum* to product ID *prodid*.**

Explanation: A part is listed in the component section of the product parameter file with a part handler EXEC that cannot be found on an accessed minidisk.

System Action: The procedure continues with RC = 4.

User Response: If this was a part that was intentionally bypassed, then no action is required. Otherwise, find the part listed in the component section of the product parameter file and verify the listed part handler spelling. Check the minidisk accesses for correctness.

1826W **The part type *fn ft* was not found while executing the *name* part handler.**

Explanation: A part listed in the component section of the product parameter file cannot be found on an accessed minidisk.

System Action: The procedure continues with RC = 4.

User Response: Find the part listed in the component section of the product parameter file and verify that the part is spelled correctly. Verify the minidisks are accessed correctly.

1827S **The *name* part handler failed for part *fn ft* while applying PTF *ptfnum* for product ID *prodid*.**

Explanation: The part handler listed in the message text failed.

System Action: The procedure exits with RC = 100.

User Response: Determine the cause of the part handler failure by examining its error messages.

1828W **The prefix *xxx* is listed in the control file *fn*, but a PTF number cannot be found.**

Explanation: A prefix (*xxx*) has been listed on a control file level indicating that the text deck will be named by a PTF number, but an auxiliary control file containing a valid PTF number cannot be found.

System Action: RC=4.
Text deck is named according to the control file without the use of a PTF number.

User Response: Check the control file and auxiliary control structure.

1829W **The *fn ft* update cannot be applied.**

Explanation: The VMFAPTXT part handler failed while attempting to apply the update listed in the message text.

System Action: The procedure continues with RC = 4.

User Response: Examine previous messages to determine the cause.

1830E ***label* has been specified as the target for the component *compname* of product *prodid*. There is no target listed on the *label* tag.**

Explanation: A symbolic string has been specified as a target, but the label in the minidisk assignments (MDA) section of the 56643082 $PPF file does not contain any minidisk address.

System Action: The procedure exits with RC = 100.

User Response: Correct the MDA section in the file and reinvoke the procedure.

1831W **PTF *ptfnum* is included in *fn ft*. It is in the exclude list *fn ft*, but cannot be excluded.**

Explanation: The PTF specified in the $EXCLIST is contained in a part of the PTF being applied. VMFAPPLY can only exclude PTFs serially from a text deck (the last PTF applied).

System Action: The procedure continues with RC = 0.

User Response: Refer to *VM/XA SP Installation and Service* for instruction on how to process PTFs that could not be excluded by VMFAPPLY.

1832W **Text deck *fn ft* is included in the *name* build list but cannot be found.**

Explanation: A text deck, listed in the build list to be included in the build, cannot be found on an accessed minidisk.

System Action: The procedure continues with RC = 4.

User Response: Check that the correct minidisks are listed in the component section of the product parameter file. Verify that this deck was intentionally removed from the access order.

1833W **Text deck *fn ft* does not contain any executable code but is listed in the *name* build list. The build process might not be complete.**

Explanation: A text shell (a text deck that contains only requisite information) has been found. It is listed in the prologue of another text deck as an update.

System Action: The procedure continues with RC = 4.

User Response: Investigate why this text deck is missing.

1834W **The file type of *fn ft* might be incorrect due to preferred AUX files.**

Explanation: The file type of a text deck was found on a level of the control file that also contains preferred files.

System Action: The procedure continues with RC = 4.

User Response: Check the control file named in the component section of the product parameter file for correctness.

1835E **AUX file *fn auxft* on *vdev* could not be saved by XEDIT when adding update *fn update-ft*. PTF(s) included in *fn* might be partially applied.**

Explanation: VMFAPPLY could not completely process all the changes included in the text deck that is a part of the PTF being applied.

System Action: The procedure exits with RC = 100.

User Response: Add another APPLY minidisk to the $PPF file or increase the size of the current APPLY minidisk and restart VMFAPPLY.

1836E **This program requires file mode A to be accessed as R/W.**
[Access mode A as R/W and re-try.]

Explanation: The procedure cannot continue without a read/write file mode A.

System Action: The procedure exits with RC = 100.

User Response: Access a read/write file mode A and restart the procedure.

1837W *fn ft fm* **update entries do not match the control file for update** *fn ft.*

Explanation: While checking the entries in the text deck with the AUX entries in the AUX file and update entries in the control file, a mismatch has been detected.

System Action: The procedure continues with RC = 4.

User Response: Investigate the problem with textdecks not matching the AUX file and control file.

1838E **The service exec requires file mode C to be accessed as R/W. Access mode C as R/W and re-try.**

Explanation: For a product with its own service EXEC, a read/write file mode C must be accessed.

System Action: The procedure exits with RC = 100.

User Response: Access a read/write file mode C and either copy the existing SERVICE DISKMAP for file mode A to file mode C, or erase any on file mode A before reinvoking VMFREC.

1839W **Multiple update files for product ID** *prodid* **have been found on:** *vdev*

Explanation: Duplicate update files exist on different minidisks for the same product.

System Action: The procedure continues with RC = 4.

User Response: Be aware that, with multiple updates, there is a possibility that the wrong decks can be picked up at build time.

1840W **The AUX file entries in** *fn1 ft1 fm1* **and** *fn2 ft2 fm2* **do not match.**

or

fn ft fm **contains AUX entries but an AUX file has not been located.**

Explanation: While comparing AUX entries between the given text deck and AUX file or between the two given AUX files, a mismatch has been detected. The PTF that was being processed when the error was detected may be partially applied.

System Action: The procedure continues with RC = 4.

User Response: Investigate the reason for the mismatch, correct it, and retry the procedure.

1841W **The** *fn ft* **update file was not found. An update shell has been created.**

Explanation: An update file pointed to by an entry in a text deck prologue or an AUX file could not be found on an accessed minidisk. Update shells are only created for text decks. When the update file is for another type of part (for example a MACRO or an EXEC), the second part of this message is not issued and no update shell is created.

System Action: The procedure continues with RC = 4.

User Response: Investigate why the update file listed in the message text was not found.

1842W **The** *fn ft* **file was not found. The exclude option was ON but there is no exclude list. The process is continuing with the exclude option set OFF.**

Explanation: VMFAPPLY cannot locate the $EXCLIST file for the control file specified in the product parameter file.

System Action: The procedure continues with RC = 4.

User Response: Investigate why the $EXCLIST file listed in the message text was not found.

1843W **The *fn ft* update cannot be applied for PTF *ptfname*. *vdev* must be R/W.**

Explanation: The update files are marked "applied" by renaming from file mode number 1 to 5. The disk that the files reside on must be read/write in order to rename.

System Action: The procedure continues with RC = 4.

User Response: If the update file should have been applied, then move the file to a read/write minidisk and restart the procedure.

1844W **For product *prodid*, PTF *ptfnum* part *fn ft* was not found.**

Explanation: A part listed in a PTF part list could not be found.

System Action: The procedure continues with RC = 4.

User Response: Investigate why the part was not found.

1845W ***fn ft* file is a text deck shell for PTF *ptfnum*.**

Explanation: A text deck that is listed as a part of the PTF listed in the message text could not be found.

System Action: The procedure continues with RC = 0.

User Response: Investigate why the deck was not found.

1846W **Update file *fn ft* has a requisite of *ptfnum*. The update file for part *part-type* cannot be found on a R/W disk. The DEPEND entry cannot be entered.**

Explanation: A text deck contains a requisite that points to an update file that cannot be found, or a depend entry cannot be written because the update file is on a R/O disk. The depend function is bypassed.

System Action: The procedure continues with RC = 4.

User Response: Investigate why the requisite update file was not found.

1847W ***ptfnum1* is referenced in update file *fn ft* and is a part of PTF *ptfnum2*. *ptfnum1* is a requisite for another component.**

Explanation: This is a warning message, indicating that a requisite exists that is outside of the component.

System Action: The procedure continues with RC = 0.

User Response: None.

1848E ***fn ptf-ft* on *vdev* could not be copied/renamed to *base-ft* on *vdev*. AUX files and update files show all service in text deck *fn ptf-ft* has been applied.**

Explanation: VMFAPPLY could not complete processing a text deck that is part of the PTF. The text deck being processed needs to be renamed to its base file type, and copied to the minidisk identified in the message. This text deck is not supported by VMFBLD.

System Action: The procedure exits with RC = 100.

User Response: Copy and rename the text deck identified in the message to another minidisk listed for the component in the $PPF file, or increase the size of the current DELTA minidisk and copy/rename the text deck.

1849W **APAR *aparnum* is a requisite of APAR *aparnum* and the aux entry *update-file type* is not included in the self documenting prolog information of the text deck *file name file type*.**

or

APAR *aparnum* is a required requisite, but part *file name file type* on disk *address* has not been applied by VMFAPPLY and the update file is not fm5.

Explanation: First message: An update file was found for the text deck listed in the message and this update file was not included in the self documenting prolog information of that text deck.

Second message: An update file was found for the APAR listed in the message and this update file was not applied by VMFAPPLY.

System Action: The procedure continues with RC = 4.

User Response: First message: Investigate why the aux entry identified in the message is missing from the updated text deck. The text deck may have been built without a requisite APAR.

If this text deck was built by IBM, this text deck is in error. Report this to your IBM service representative.

If the text deck was built locally, you may have left out a required entry in an aux file for the text deck identified in the message. Correct the aux file and run VMFBLD again.

Second message: Investigate why this update file has not been applied.

1850I **The processing for *part-name* by the *fn* EXEC was bypassed.**

Explanation: Processing of a part has been bypassed. This could occur if the part handler has been commented out in the product parameter file.

System Action: The procedure continues.

User Response: If the part was expected to be handled, investigate why the part handler EXEC was not found.

1851I **Processing *fn ft* with the part handler *parthandler* EXEC.**

Explanation: This is an informational message indicating that the function requested is progressing to the next part.

System Action: The procedure continues.

User Response: None.

1852I **This is *volume-number* of *total-volumes*, level *level* {*PUT|COR*} tape.**

Explanation: This is an informational message indicating the level of the tape being processed.

System Action: The procedure continues.

User Response: If the tape level is correct, no response is necessary. Otherwise, stop the procedure, mount the correct tape, and restart the procedure.

1853I **Processing PTF *ptfnum*.**

Explanation: This is an informational message indicating that the function requested is progressing to the next PTF listed in the PTF parts file.

System Action: The procedure continues.

User Response: None.

1854W ***ptfnum* is a requisite for another component.**

Explanation: This message indicates that a PTF has been found that lists a requisite that is not a part of the component being processed.

System Action: The procedure continues with RC = 4.

User Response: Verify that the out-of-component requisite will be applied.

1855W **Control file *fn* contains invalid data. Text deck *fn* will not be named by PTF number.**

Explanation: A record was found in the control file that violates the control file rules as defined in *VM/XA SP CMS Command Reference* under the UPDATE command.

System Action: RC=4.
The text deck is named according to the control file without the use of a PTF number.

User Response: Check the control file and AUX structure for invalid level IDs or update identifiers.

1856I **PTF *ptfnum* is in the *fn ft* exclude list for product ID *prodid* , and will not be applied.**

Explanation: This is an informational message indicating that a PTF listed in the apply list will not get applied, because it is also listed in the exclude list.

System Action: The procedure continues.

User Response: None.

1857I **APAR** *aparnum* **is a required requisite, but part** *fn1 ft1* **on disk** *disk address* **has not been applied by VMFAPPLY and the update file is not fm5.**

Explanation: A textdeck includes a requisite which has an update that has not been applied as indicated by a file mode other than fm1.

System Action: The procedure continues with return code 0.

User Response: The message is informational. The part identified in the message text is a part that must be processed manually. VMFAPPLY changes the file mode of update files for TEXT decks to a 5 when processed. Update files for other parts, such as EXECs, XEDIT macros, MACROs and COPY files, will force this message to be issued since file modes on update files for these parts are 1 (VMFAPPLY doesn't process these update files). The message can be eliminated by changing the file mode on update files for these other parts to a 5. Next time VMFAPPLY is run (you don't have to run VMFAPPLY again now), these messages will not be issued.

1858I *message*

Explanation: The variations of this message are explained below.

System Action: In each case, the procedure continues.

User Response: In each case, refer to *VM/XA SP Service Guide* for instructions on how to regenerate service parts of the system.

Messages:

- *fn* **build list contains files that have been serviced.**

  Explanation: VMFAPPLY has processed a PTF that contained a new level of a file contained in the build list found in the build (BLD) section of the product parameter file. VMFBLD should be invoked to regenerate the object associated with the build list, such as the nucleus and the module.

- *fn ft* **has been serviced.**

  Explanation: VMFAPPLY has processed a PTF that contained a new level of the file identified in the message. A build step might be required to make the change available for execution.

1859I **The** *name* **build exec completed with return code** *rc*.

Explanation: The part handler identified has completed the build step required for a build list specified in the product parameter file.

System Action: The procedure continues, with return code set to *rc*.

User Response: None.

1860E *compname* **is an invalid component name for product ID** *prodid*.

Explanation: The base *compname* listed on the override compname tag was not found in the complist tag.

System Action: The procedure exits with RC = 100.

User Response: Verify that the override and complist tags match. If not, change the tag that is incorrect.

1860I *compname* **is an invalid component name for product ID** *prodid*.

Explanation: A component name has been entered for a product that does not contain that component.

System Action: The procedure continues and prompts you for a valid *compname*.

User Response: Respond to the prompt by entering a valid *compname*.

1861W **The** *fn* **service exec completed with return code** *rc message*.

Explanation: The service EXEC has completed the task. The return code (*rc*) indicates whether the exec was successful or not at completing the task. *message* may be one of the following according to the return code (*rc*) specified:

- *rc* = 0

The service has been received.

VMFREC has invoked a product service EXEC, and the EXEC has completed successfully.

- $0 < rc > 4$

  Service received, might have been done previously.

  VMFREC has invoked a product service EXEC, and the EXEC has completed successfully. Service for this EXEC may have already been received before.

- $3 < rc > 8$

  Service received, a build might be required.

  VMFREC has invoked a product service EXEC, and the EXEC has completed successfully. The product for which service was received may need to be rebuilt in order to apply the service.

  User Response: Review service procedures for the product service to check if a build is required.

- $7 < rc > 12$

  Receive has been discontinued, some service may have been received.

  VMFREC has invoked a product service EXEC and the EXEC was halted before service was completely received.

  User Response: Correct the problem that caused the product service EXEC to fail and restart VMFREC.

- $rc > 11$

  Receive has been discontinued, tape position unknown.

  VMFREC has invoked a product service EXEC and the EXEC has been halted at an unknown point in the process.

  User Response: Correct the problem that caused the product service EXEC to fail and restart VMFREC.

System Action: VMFREC exits with *rc* unless the list option was specified, in which case the next product in the list is processed.

User Response: None, unless otherwise indicated above.

1862I The LTO option was specified, but not all targets are accessed R/W as required. The LTO option has been set OFF.

Explanation: In order to specify the last-text-only option, the target minidisks must be accessed as read/write.

System Action: The procedure continues with the last-text-only option turned off.

User Response: If the last-text-only option is desired, stop the procedure and reaccess the target minidisks in write mode, then restart the procedure.

1863W Service has been received on the S minidisk. Module generation and/or reIPL of CMS might be required.

Explanation: Service has been received onto the system minidisk.

System Action: RC=4.

User Response: The system minidisk needs to be re-IPLed, or the CMS segment might need to be resaved.

1864E *Service exec* **previously failed for** *prodid compname1* **on PUT** *level1*
You must rerun *service-exec* **for** *prodid compname1* **on PUT** *level1* **before running** *service-exec* **for** *prodid compname2* **on PUT** *level2*

Explanation: More than one $ER* erase list has been found in the access order. Update files from the level 1 PUT were not processed completely.

The variables in the message text are defined as follows:

service-exec	The main EXEC name (VMFREC)
prodid	The product ID
compname1	The previously failing component ID
level1	The failing PUT level
compname2	The current component ID
level2	The current PUT level

System Action: RC=23 (for merge error). The function is terminated.

User Response: The function cannot be reinvoked until one of the duplicate $ER* erase lists is removed from the access order. The component

identified in the message text should be run after the duplicate file is removed.

1865W **Update files for *prodid compname* have been found on the target. The target should be empty. The MERGE tag parameter in the $PPF file might be improperly specified.**

Explanation: The target minidisk should be empty at the start of receive for update files. If the target minidisk is listed on the MERGE tag in the PPF correctly, the minidisk should be empty.

System Action: RC=4.
A prompt allows the user to continue with the process or discontinue receive.

User Response: Unless the intent is to receive updates to a minidisk that has already received updates, the user should exit and check the PPF for correct string labels on the MERGE tag.

1866W **Duplicate update files have been found in the target and APPLY access orders. Merge might not have been done correctly.**

Explanation: Duplicate update files have been found on the target and APPLY minidisk strings.

System Action: Update file processing continues with RC = 4.

User Response: With the proper use of the service tools and MERGE tag in the PPF, this situation should not occur. Manual file manipulation is necessary in order to remove the duplicate files.

1867E **The loadlist *loadlist name* EXEC [*vdev|dirname*] does not contain a valid loader name.**

Explanation: The loader invoked for the build of the component being processed is determined by the loader card specified in the loadlist identified in the :BLD. section of that component in the product parameter file. This loadlist does not contain a loader card that is recognized by the EXEC as a valid loader.

System Action: The procedure exits with RC = 100.

User Response: Examine the loadlist for a loader card. If one does not exist then add one. If one does exist then it is not supported by the parthandler invoked to process it. Modify the product parameter file to invoke a parthandler that does support the format of this loadlist.

1868W **AUXFILE *fn ft* contains PTF# *ptfnumber* but textdeck *fn ft* could not be located. Search will continue with the level ID from the control file.**

or

The control file *fn* CNTRL indicates that part *fn2 ft2* should exist but it can not be located. Searching will continue at the next level.

Explanation:

First message: In determining the file type of a textdeck using the AUX and control file structure, an AUXFILE has been found that contains a valid PTF or local tracking number. A textdeck file type formed with this number and prefix cannot be located using the current access order. This may result in a build being done with a lower level textdeck.

Second message: In determining the file type of a textdeck using the AUX and control file structure, an update or AUX file was specified in the control file, and the update or AUX file exists on an accessed disk but a part with a file type of the prefix concatenated with the LEVELID could not be located.

This condition may also exist if the update specified in the control file contains a prefix but the next AUX level points to an AUXFILE that does not contain a valid PTF number. In this case, a search is made using the LEVELID from the control file which may cause the message to be issued as described above.

System Action: In each case, the procedure will continue to process the control file searching for a textdeck file type. RC is set to 4.

User Response: In each case, investigate the access order to determine if the textdeck identified in the message can be located. If it can be located, either the disk should be added in the PPF, or the textdeck copied to a disk that is listed in the PPF. If the deck cannot be located, then the module should be re-assembled.

1869R *prodid compname* **begins on** *PUT\|COR tapenum* **volume** *vol.* **Mount volume** *vol* **and press ENTER or type QUIT.**

Explanation: The service you are receiving for a product/component begins on another tape.

System Action: The procedure continues when you press ENTER, or exits if you type QUIT.

User Response: Mount the tape containing the service for the component and press ENTER. Alternatively, type QUIT, mount the next tape and reinvoke VMFREC to load the rest of the service for the component.

1870R **Choose the component which is to be processed from the list below:**

Explanation: A procedure has been requested, but a component name was not specified. The names displayed are taken from the COMPLST and OVERLSTP tags in the product parameter file.

System Action: The system waits for a response to the prompt.

User Response: Enter a component name from the list displayed, or enter QUIT.

1871R **Do you want to continue: (YES/NO).**

Explanation: An opportunity is given to prematurely exit the procedure.

System Action: The system waits for a response to the prompt.

User Response: Answer the prompt.

1872E **COR is a nonsupported option for the** *prodid* **product ID. The tape has been positioned to the beginning of the product's first file. Use the product's corrective service procedure to process.**

Explanation: The product specified on the VMFREC command is not packaged in the format required by VMFREC. Refer to the product's documentation for the procedure to follow for corrective service.

System Action: The procedure exits with RC = 8.

User Response: None.

1873W *prodid compname* **relative tape file** *number, label* **is listed in the product directory but is not in the $PPF file. This tape file will not be received.**

Explanation: The product specified on the VMFREC command is not packaged in the format required by VMFREC. Refer to the product's documentation for the procedure to follow for corrective service.

System Action: The procedure continues with RC = 4.

User Response: Correct the receive service (RECSER) section in the $PPF file for the component specified on the VMFREC command. Add the label and part handler for the tape file.

1874R **The** *execname* **option** *PUT\|COR* **does not match the type of the mounted tape** *PUT\|COR.* **Do you want to continue? (YES/NO)**

Explanation: VMFREC is receiving service from a preventive or corrective tape, but the option specified on the command does not match the tape type.

System Action: The procedure exits if response is NO, continues if response is YES.

User Response: Enter YES or NO.

1875E **XEDIT cannot save file** *fn ft* **on disk** *fm.* **The return code from XEDIT was** *rc.* **Correct the problem and re-start the procedure from the beginning.**

Explanation: The procedure being run could not save the file being edited. The minidisk might be full.

System Action: The procedure exits with RC = 100.

User Response: Look up the return code from XEDIT and determine the cause of failure. Correct the problem and restart the procedure from the beginning.

1876E **The VMFAPPLY option *PUT|COR* does not match the apply list specified.**

Explanation: VMFAPPLY was invoked with the PUT or COR option, but the $APPLIST being used does not match the option specified.

System Action: The procedure exits with RC = 100.

User Response: Reinvoke VMFAPPLY, using the correct option.

1877R *prodid compname* **begins on** *PUT|COR tapenum* **volume** *vol.* **Do you want to continue? (YES/NO)**

Explanation: VMFREC is receiving service for a component that started on a previous tape. You should not receive this service if you have not already received the service from the other tape.

System Action: The procedure exits if response is NO, continues if response is YES.

User Response: Enter YES or NO.

1878I **No files have been found for** *prodid compname* **on the** *PUT|COR* **level** *level.*

Explanation: VMFREC could not find any service for the product/component on the tape that is currently mounted.

System Action: The procedure continues with RC = 0.

User Response: Check other tape volumes for this service level for the product/component.

1879R *product-name component-name* **on** *PUT|COR level-number* **volume** *vol-number* **continues on volume** *vol-number2.* **Mount volume** *vol-number2* **and press ENTER or type QUIT.**

Explanation: The service you are receiving for a product/component is continued on another tape.

System Action: The procedure continues when you press ENTER, or exits if you type QUIT.

User Response: Mount the tape containing the remainder of the service for the component and press ENTER. Alternatively, type QUIT, mount the next tape and reinvoke VMFREC to load the rest of the service for the component.

1880W *fn ft* **is an update file that has a requisite for APAR** *aparnum,* **but no update file can be found.**

Explanation: VMFAPPLY found a requisite APAR for the update being applied, and could not locate an update file that contained the APAR number in the file type. The file type of the update file being applied could not be added to the DEPEND tag for this update.

System Action: The procedure continues with RC = 4.

User Response: Locate the update file(s) for the APAR, and add the file type specified in the message to the DEPEND tag. The situation where no update files can be found is not an error condition if the PTF that corresponds to the given *ft* does not contain any parts that are serviced with update files.

1881W **Text deck** *fn ft* **on** *vdev* **contains an applied update, but the AUX file cannot be found on the APPLY string. The PTF was not applied.**

Explanation: The text deck being processed contains an update that has a file mode number of 5, which indicates it has been added to an AUX file. VMFAPPLY could not find an AUX file on any minidisk listed on the APPLY tag in the $PPF file.

System Action: The procedure continues with RC = 28.

User Response: Change the file mode number of the update file to a 1 and reinvoke VMFAPPLY, or create an AUX file for the text deck on a minidisk listed in the APPLY string, and add the update file.

1882W **AUX file** *fn ft* **on** *vdev* **is inconsistent with the text deck** *fn ft* **on** *vdev.* **The PTF was not applied.**

Explanation: The text deck being processed contains an update that has been applied, but it is not contained as the first entry in any AUX file found in the APPLY string. VMFAPPLY cannot determine where the update being applied should be added to the AUX file.

System Action: The procedure continues with RC = 4.

User Response: Change the file mode number of the update file to a 1 and reinvoke VMFAPPLY, or

add the update to an AUX file for the text deck on a minidisk listed in the APPLY string.

1883W **There was a problem loading the apply and exclude lists.**

Explanation: VMFREC was not able to copy/rename the apply and exclude contained on the service tape to the target minidisk. The minidisk might be full, or an incorrect level of the apply or exclude list might have been shipped on the tape.

System Action: The procedure continues with RC = 28.

User Response: Check the target minidisk for the correct level of the apply and exclude list. If found, then copy/rename them to $APPLIST and $EXCLIST. If they are not found, contact your IBM support center.

1884W *ptfnum* **is an included|dependent PTF in** *fn1 ft1*. **It has been added to** *fn2 ft2* **apply list.**

or

ptfnum **is a dependent PTF in** *fn ft*. **It has been added to the** *control-file name* **$APPINCL apply list.**

Explanation: While applying a PTF, a text deck or AUX file was processed that contained additional PTFs that were not in the $APPLIST. VMFAPPLY will process these PTFs as if they were in the $APPLIST, to ensure that all service required for the PTF is applied.

System Action: The procedure continues with RC = 0.

User Response: None.

1885W *fn1 ft1* **listed in** *fn2* **CNTRL represents service at a higher level than** *fn3 ft3*. **The higher-level service may need to be reworked or removed.**

Explanation: VMFAPPLY found an AUX file or update file at a higher level in the control file than the AUX file being updated by VMFAPPLY. If the file found is an AUX file, its contents follow the message.

System Action: The procedure continues, with RC = 0.

User Response: The higher-level service must be investigated to determine if it conflicts with the new service in the AUX file being updated by VMFAPPLY. Any service that is duplicated by the new service must be removed from the higher level. Any service that conflicts with the new service must be reworked so that it no longer conflicts.

1886W **Update file** *fn ft* **on** *addr* **has a file mode other than fm5 but auxfile** *fn ft* **on** *addr* **contains an AUX entry for this update. The PTF was not applied because of this inconsistency.**

Explanation: During apply processing of a text deck, the deck is read from bottom up looking for AUX entries. When an AUX entry is found the update file type is taken from the first field and a check has been made of the update file mode in order to determine if the update has been applied. The update has been found with a file mode other than 5 indicating it is not applied. The AUX file was then checked and an entry for this update has been located indicating the update has been previously applied.

System Action: Processing by the current parthandler will end resulting in the PTF not being applied. RC is set to 4.

User Response: Investigate if duplicate updates exist that have already been applied, indicated by having a file mode 5. If so, the current update can be renamed to fm5 or erased. If a duplicate is not found then the update should be renamed to fm5. VMFAPPLY can then be used to reapply this PTF or it can be applied manually.

This message may indicate that a PTF has incorrectly been unapplied. It also may indicate that update files have been received by a part handler other than VMFRCUPD.

1887W **The** *fn* **EXEC file was not found. The {user exit|part handler}** *fn* **was not executed.**

Explanation: The specified EXEC was not found.

System Action: The procedure continues without executing the EXEC.

User Response: If the EXEC was needed for processing, ensure that the EXEC is available to be executed, and restart the procedure.

1888W **Product parameter file *prodid* $PPF has been found on file mode *fm*. It has been renamed to *prodid* $PPFSAVE *fm*. A new product parameter file will be loaded from the tape. Merge any local changes from the renamed file to the new file.**

Explanation: If you had a copy of the product parameter file on the target disk for VMFREC, it would be overlaid. A copy is saved as file type $PPFSAVE and a new $PPF is loaded down to the target disk for VMFREC.

System Action: The procedure continues.

User Response: Merge your changes from the $PPFSAVE file into the new product parameter file loaded down to the target disk.

1889E **The *prodid* $PPF product parameter file contains invalid data on the *ppftag* record Correct the file and retry.**

Explanation: An error has occurred because of invalid data contained on the product parameter file record indicated by the message text.

System Action: The procedure exits with RC = 100.

User Response: Correct the invalid data, and restart the procedure.

1890E **The *ppfname* $PPF product parameter file is invalid. *record* is missing or out of order. Correct the file and retry.**

Explanation: A required record cannot be found in the product parameter file.

System Action: The procedure exits with RC = 100.

User Response: Add the required record, and restart the procedure.

1891E **Part handler *name* failed for component *compname* of product ID *prodid*.**

Explanation: A part handler EXEC that is listed in the product parameter file for the component indicated by the message text has failed.

System Action: The VMFBLD procedure exits with RC = 100. VMFREC will continue with any other *prodid* components, if the list option was specified.

User Response: Correct the problem that is causing the part handler EXEC to fail, and retry the procedure.

1892E **The *name* build list was not found in the *ppfname* $PPF file.**

Explanation: A build list name is missing.

System Action: The procedure exits with RC = 100.

User Response: Add a build list name to the PPF.

1893E **The *name* tag found in apparent $PPF override file *ppfname*.**

Explanation: A tag was found in a $PPF override file that was not expected. The most likely possibility is that it is a COMPLST tag.

System Action: The procedure exits with RC = 100.

User Response: Correct the $PPF file and restart the procedure.

1894W **Ensure that all parts of *prodid compname* have been received before proceeding to the APPLY step.**

Explanation: In order to do a complete APPLY, it is necessary that the entire component be received. This message is issued if part of the component has not been received during this invocation of VMFREC.

System Action: The procedure continues, but will exit with RC = 4.

User Response: Receive the rest of the component, if you have not already done so.

1895E ***name1* tag found** {*before*|*after*} ***name2* tag in *ppfname* $PPF product parameter file.**

Explanation: A tag was not in the proper place in the $PPF file.

System Action: The procedure exits with RC = 100.

User Response: Correct the $PPF and restart the procedure.

1896E **The component name is missing from the component override tag *name*.**

Explanation: An override was specified, but no base component was listed in the override $PPF file.

System Action: The procedure exits with RC = 100.

User Response: Correct the $PPF file, and restart the procedure.

1897E **Error reading file *fn ft fm*.**

or

A problem occurred reading file *fn ft*.

Explanation: A file required by the procedure cannot be read.

System Action: The procedure exits with RC = 100.

User Response: Correct the problem with the file and retry the procedure. Some possible problems may be: not enough storage, or bad file pointers.

1898E **A record can not be written to file *fn ft fm*.**

Explanation: A problem has been encountered in trying to add a record to a file.

System Action: The procedure exits with RC = 100.

User Response: Correct the problem with the minidisk and retry the procedure. The most likely cause of this problem is that the minidisk is full.

1899E **There is a problem loading a file.**

Explanation: A problem has been encountered writing a file.

System Action: The procedure exits with RC = 100.

User Response: Correct the problem with the minidisk and retry the procedure. The most likely cause of this problem is that the minidisk is full.

1900W **The existing *ppfname* $SETUP A1 file has been refreshed. You might want to check your access order when done.**

Explanation: VMFSETUP has been invoked for accessing two consecutive times, without being invoked with the RESTORE option. The first access order has been lost.

System Action: The procedure continues with RC = 4.

User Response: Check your access order at the completion of processing.

1901E **VMFSETUP, when invoked with the PPFTEMP parameter, expects a temporary product parameter file. *ppfname* $PPFTEMP was not found.**

Explanation: VMFSETUP was invoked with the PPFTEMP operand. This operand is only to be used if VMFOVER has been invoked previously to create a $PPFTEMP file. Because no $PPFTEMP file was found, VMFOVER could not have run successfully.

System Action: The procedure exits with RC = 28.

User Response: Invoke VMFOVER before executing VMFSETUP, or invoke VMFSETUP without the PPFTEMP operand.

1902E **The *compname* component name was not found in the *ppfname* $PPFTEMP file.**

Explanation: The $PPF file does not contain a section for the specified component.

System Action: The procedure exits with RC = 100.

User Response: Correct the $PPF file, and restart the procedure.

1904E **The current access order does not match the access order that is in the RELEASE section of *ppfname* $SETUP. Your access order will be left as it is.**

Explanation: VMFSETUP cannot restore the original access order, because the current access order is not the same as the one set up by VMFSETUP when it was originally invoked.

System Action: The current access order is left untouched, and the procedure exits with RC = 100.

User Response: Check your access order.

1905W **The access of *vdev* failed with a return code of *rc*. Processing continues for product *prodid*.**

Explanation: VMFSETUP was not able to perform an access that was listed in the $PPF file. The minidisk may not have been linked or formatted.

System Action: The procedure continues with RC = 4.

User Response: Ensure that the access was not needed. If it was needed, you should make the minidisk accessible, and then restart the procedure.

1906E **The access of *vdev* failed with a return code of *rc*. Processing stops for product *prodid*. [The original access order cannot be restored.]**

Explanation: VMFSETUP was not able to perform an access that was listed in the $PPF file. The minidisk might not have been linked or formatted. It has been determined that this minidisk might be essential for processing.

System Action: The procedure exits with RC = 100.

User Response: Take whatever steps are needed to allow the minidisk to be accessed, then restart the procedure.

1907I **Assembling *fn* [(*options*)].**

Explanation: The assembly is going to begin. If you specified any assembler options, the options used are displayed.

System Action: The assembly begins.

User Response: None.

1908E **Error {assembling|updating} *fn*.**

Explanation: If assembling is taking place, an assembler error occurred. If updating is taking place, a severe update error occurred.

System Action: Execution is terminated.

User Response: Correct the error and rerun.

1909I ***fn ft*** **A [has not been] created.**

Explanation: A text file with the given file name and file type has been created, or the text file was not produced because of assembler errors.

System Action: None.

User Response: None.

1911E **Invalid CSECT name *csect name* for ICS card in *fn ft*.**

Explanation: The include control section card has an invalid CSECT name specified.

System Action: The procedure exits with RC = 100.

User Response: Correct the ICS card, and restart the procedure.

1912E **VER card missing in *fn ft*.**

Explanation: A VER card was missing, and the patch cannot be performed.

System Action: The procedure exits with RC = 100.

User Response: Add a VER card, and restart the procedure.

1932E **Part-handler *execname* for build list *listname* was found in the *fn* $PPF but a target was not specified.**

Explanation: The name of the target string for the specified part-handler and build list in the BLD section of the $PPF file is missing.

System Action: The procedure ends with RC=100.

User Response: Correct the $PPF file and reenter the command.

1937I **Merge processing has started for *fn1 fn2*.**

Explanation: The merge tag in the product parameter file contains at least one set of disk strings. It has been determined that the component being received has not been previously received and that at least one of the first disks of the disk strings were not empty.

System Action: The procedure continues with the merge of the disk strings. For each symbolic disk string listed on the merge tag, the contents of each

disk is moved to the next, starting with the last pair in the symbolic disk string.

User Response: No response required.

1938I The merge of *symbolic-disk-string-name - dsk1* to *dsk2* is complete.

Explanation: As part of merge processing, files from each disk listed in a symbolic disk string are moved to the next disk starting with the last pair of disks. The contents of *dsk1* have been successfully copied to *dsk2* as a result of processing the symbolic disk string indicated in the message.

System Action: The procedure continues merging with the next pair of disks in the symbolic disk string. If no other pairs exist then if more symbolic disk strings exist on the merge tag, they are processed. Otherwise merge is complete.

User Response: No response required.

1939I Merge processing is complete for component *compname*.

Explanation: As part of merge processing, files from each disk listed in a symbolic disk string are moved to the next disk starting with the last pair of disks. This process is repeated for each symbolic disk string listed on the merge tag in the product parameter file. This message indicates that all symbolic disk strings have been merged.

System Action: The procedure continues receiving service for the component specified.

User Response: No response required.

1940I Merge processing is not required for component *compname*.

Explanation: The first disk of each symbolic disk string listed on the merge tag in the product parameter file has been found to be empty, making a merge unnecessary.

System Action: The procedure continues receiving service for the component specified.

User Response: No response required.

1941E Component *compname* has already been received successfully from *PUT|COR tape-identifier*.

Explanation: When a tape has successfully been received a merge indicator file is written to the first disk of each symbolic disk string listed on the merge tag in the product parameter file. This merge indicator has a file name of PUT or COR depending on tape type and a file type of "$MR" concatenated to the first letter of the tape type - P or C, concatenated with the tape identifier. Another indicator file with a file name of "$" concatenated with the tape type and a file type of the tape identifier is created with the comment in the COR or PUT DOCUMENT which contains a date and time stamp. The tape identifier from the "$MR" file is compared to the tape type currently being received. If both of these are identical then the comment in the second indicator file is compared with the comment from the current COR or PUT DOCUMENT. If both of these comments are identical then this message is issued.

System Action: The procedure exits with RC = 100.

User Response: This message indicates that the service from this tape has already been successfully received. Check that the correct tape has been mounted. Also verify that the correct component has been entered. If the real intent is to re-receive service for this component then the indicator files described above can be renamed or erased so that it appears a receive has not been completed.

1942I Service has been received for *partname parttype*. It is a part of the Serviceability Enhancements.

Explanation: A Serviceability Enhancement part has received service. This service includes a new executable version for *partname*.

System Action: The procedure continues.

User Response: The executable part shipped for *partname* may be at a lower service level than your existing *partname*. Review the self-documenting information at the end of EXECs, XEDIT macros, and modules to determine the service levels of the new and existing parts. If necessary update the executable version used on your system to the latest level. Copy the PTF numbered version of the part to the executable version.

Example:

```
COPY VMFAPCOM EXC12345 B VMFAPCOM EXEC B
```

1943W **One or more AUX files have been received for manually supported parts. The AUX files for these parts must be evaluated and may need updating to assure the validity of your system.**

Explanation: The AUX file(s) that have been received contain AUX entries for all updates to the part being serviced from the base level of the product to the update that was received. Some of the AUX entries in the new AUX file may already appear in other AUX files on your system. Having the same AUX entry appear more than once on your system can affect your system's integrity.

System Action: The procedure continues with RC = 4.

User Response:

- Using the $VMFREC $HISTORY file that was created during execution of VMFREC EXEC, make a list of all of the AUX files that were received. The $VMFREC $HISTORY contains entries from all runs of VMFREC EXEC. The entries from the latest run are on the bottom and are outlined with beginning and ending time stamps.
- For each AUX file received, use the LISTFILE command to make a list of all of the existing AUX files for this part.
- Use XEDIT to view CNTRL file that is used for the given part.
- Identify the appropriate AUX level in the CNTRL file for the given part. For example, if you received corrective service and there is an AUXCOR level in the CNTRL file, then this would be the appropriate level for the new AUX file.
- If necessary, rename the new AUX file so that its file type matches the appropriate AUX level in the CNTRL file.
- Compare the AUX entries in each AUX file level that is below the new AUX file level with the AUX entries in the new AUX file. For any duplicate entries, use an asterisk (*) to comment them out of the new AUX file.
- Compare the AUX entries in the old AUX file at the same level as the new AUX file with the new AUX file. If the old AUX file at this level contains more AUX entries, erase the new one or you will down level your system.
- Compare the AUX entries in each AUX file level that is above the new AUX file level. For any duplicate entries, use an asterisk (*) to comment them out of the higher level AUX files. If there are any Local Modifications or Local Text Patches, you must determine if they intersect with the new update and rework them if necessary.

1944E **Update file *fn1 ft1 fm1* contains an unsupported statement type *type* for the patch facility.**

Explanation: The update file identified in the message is a patch update which contains a statement other than NAME (Name for CSECT), ICS (Included control section), VER (verify), REP (replace) or comments. Only these statement types are allowed in patch update files.

System Action: The procedure continues with RC = 100. Invalid cards are not included in the text deck.

User Response: While creating the update file, you may have incorrectly typed the update statements. If so, correct the statement(s) and retry the procedure. Incorrect update files can generate a bad nucleus, therefore the update file should be corrected.

1946E **The *com* command failed with a return code *rc*. The input file specification was *fn1 ft1 fm1*. The output file specification was *fn2 ft2 fm2*.**

Explanation: The given CMS command experienced a severe error. If there is no input specification associated with the command, this part of the message is omitted. The same is true for the output specification.

System Action: The procedure continues with RC = 100.

User Response: See the given CMS command for more information.

1947E **A 3-character abbreviation could not be found for the file type *ft* in the :APP. section of *prodid prodidft*.**

Explanation: Entries in the :APP. section of a $PPF (product parameter file) have file types and file type abbreviations with which to rename the file that is being processed. If a file is being processed and its file type abbreviation cannot be found, it is an error condition.

System Action: The procedure continues with RC=100.

User Response: Verify that you are using the correct level of the PPF and that it contains the required abbreviation in the :APP. section of the component that you are servicing.

1948W **AUX file *fn ft* on *cuu* must be renamed by its file type abbreviation and a PTF number, but it does not contain a PTF number. It is being renamed to *fn ft2*.**

Explanation: AUX files that are supplied on service tapes have file types that may be listed in a control file. To avoid the new AUX file from being picked up and possibly conflicting with your system it must be renamed. The new file type is the 3-character abbreviation for the AUX file which is found in the product parameter file (PPF) concatenated with the PTF number found in the top AUX entry in the AUX file. Since the AUX file does not contain any valid PTF numbers, the new file type will be only the 3-character abbreviation.

System Action: The procedure continues with RC=4.

User Response: None.

1949W **PTF *ptfnum* for product *prodid* has already been applied.**

Explanation: A PTF was included in the apply list that had already been successfully applied. This determination was made by looking at the file mode of the $PTFPART file for this PTF. A file mode of 5 indicates that the PTF has been completely applied.

System Action: The procedure continues with RC=0.

User Response: None.

1953W ***fn* was interrupted the last time it was run. *fn* $ERRLOG has been restored from the saved history file *fn* $OLDLOG.**

Explanation: A copy of the $ERRLOG named $VMFREC, $VMFAPP, or $VMFBLD $OLDLOG is saved each time VMFREC, VMFAPPLY, or VMFBLD is invoked. If for some reason the execution of the EXEC is not completed (for example, if the user enters "HX" or the system abends) the $VMFxxx $OLDLOG remains on the users' A-disk.

Subsequent VMFREC, VMFAPPLY, or VMFBLD invocations append the $VMFxxx $OLDLOG to the new $VMFxxx ERRLOG and erase the $VMFxxx $OLDLOG. The $VMFxxx $OLDLOG will not exist after a complete run of the associated EXEC because it is eraseed, however the user should never erase the $VMFxxx $OLDLOG unless they do not want this historical exception log information.

System Action: This message has no effect on system action.

User Response: Investigate the previous messages to determine if further action is required.

1961W **The Service Enhancement parts have been received from the tape header files but they have NOT been activated.**

Explanation: VMFREC has received the service execs from the first two tape files but due to a detected inconsistency they have not been renamed to their executable form(s).

System Action: RC = 4.

User Response: Examine other messages issued to determine the nature of the inconsistency, correct it and rerun VMFREC INFO.

1962W **No PTF numbered Service Enhancement parts were found in the tape header files.**

Explanation: VMFREC has determined that no Service Enhancement EXECs or related parts exist as PTF-numbered files in tape files one or two.

System Action: RC = 4.

User Response: If PTF-numbered service EXECs were expected to be on the tape determine why they are missing. If the tape was created prior to having PTF-numbered service enhancement parts in the tape header files then no action is required.

1963I **Level *nnnn* of the Service Enhancement parts is now established on *mode cuu*.**

Explanation: VMFREC has loaded the indicated level of the service EXECs onto the reported disk and renamed them so that they are executable.

System Action: RC = 0.

User Response: None

1964E The update of the Service Enhancements part set inventory file $LEVEL EXEC was unsuccessful.

Explanation: VMFREC was unable to write the $LEVEL EXEC file to disk while creating or updating it. This file contains a list of the service EXECs and associated parts that have been installed on the users system.

System Action: RC = 100.

User Response: Ensure that the target disk [A|C] is writable and rerun VMFREC INFO.

1965E The *name* command has failed with Return Code *rc* while operating upon file *fn ft fm cuu* execution is terminated.

Explanation: The program was unable to execute the reported CMS command while operating on a CMS file.

System Action: RC = 100.

User Response: Use the reported command name, return code and file specified to determine the appropriate action to take.

1966I The *fn ft fm (cuu)* file saved as *fn ft fm (cuu)*.

Explanation: VMFREC has saved the existing service EXECs and associated parts prior to receiving a new version of these files.

System Action: RC = 0.

User Response: None

1967I The Service Enhancements part set [*on DISK mode (cuu) | in the TAPE header files*] is at the *nnnn* level.

Explanation: The level of the service EXECs and associated parts that is on the user's system is always reported. This is extracted from the $LEVEL EXEC file.
The level of the service EXECs and associated parts on the service tape is reported if it differs from the level the user has installed.

System Action: RC = 0.

User Response: If the reported level on disk differs from what is expected, determine why before proceeding.
When the level on the tape is more current than the level on disk, examine the documentation supplied with the tape to determine if the new EXECs should be received via VMFREC INFO.

1968W The Service Enhancements part set on DISK is at an UNKNOWN level

or

The Service Enhancements part set in the TAPE header files is at an UNKNOWN level.

Explanation: Either VMFREC was unable to determine the level of the service EXECs on the user's system because it could not find a $LEVEL EXEC file, or
VMFREC was unable to determine the level of the service EXECs on the service tape even though it contains some service enhancements parts in files one and two of the tape.

System Action: RC = 4.

User Response: Determine why VMFREC is unable to find the $LEVEL EXEC file. Reestablish it manually or by running VMFREC INFO.
If the tape level is unknown, request a new service tape from IBM.

1969W The Service Enhancements part set in the tape header files, specified by $LEVEL $TAPE, conflicts with the tape inventory file $LEVEL MAP.

Explanation: VMFREC has found an inconsistency between the tape's inventory of service EXEC parts and the actual parts in tape files one and two. The tape is in error.

System Action: RC = 4.

User Response: Request a new service tape from IBM.

1970W Entry for PTF *ptfno*, cannot be found in any AUX file for *file name*.

Explanation: VMFAPPLY is processing a list of all AUX file entries for the part identified in the message.

VMFAPPLY is attempting to verify that the PTF which contains service for this part exists in that list of AUX file entries but it cannot be found. (The PTF number would be in the third token of this list of AUX file entries.)

System Action: Processing continues.

User Response: This PTF is listed in the $APPLIST (apply list) but is not part of any AUX file for this part on your system. An AUX file on your system may be in error or the $APPLIST you are using may be in error. An AUX file could be in error because IBM shipped you an incorrect AUX file or you may have modified an AUX file that was sent to you from IBM.

Contact your IBM service representative for assistance.

1972W The service tape part inventory file $LEVEL MAP, is missing from the tape.

Explanation: VMFREC is unable to find a $LEVEL MAP file in tape file 1 of the service tape. Either the tape is in the old format or it is in error.

System Action: RC = 4.

User Response: If the tape was expected to have PTF numbered service parts in files one and two then order a new service tape.

1973W The service tape may be in the old format. Using INFO (ALL may back level your Service EXECs).

Explanation: VMFREC has determined that the service tape may be in the old format and using the VMFREC INFO (ALL command may receive back level executable parts onto your disk.

System Action: RC = 4.

User Response: Do not use VMFREC INFO (ALL unless you know it will not destroy your production service EXECs.

1974I A $PPF file for product *prod-id* exists on this tape volume but is not installed on your system. It must be received prior to servicing that product.

Explanation: VMFREC has checked the $PPF files in tape file two of the service tape and has determined that a $PPF file exists on the tape that is not listed in the $LEVEL EXEC file.

System Action: RC = 0.

User Response: Before servicing the indicated product ensure that its new $PPF has been installed on your system.

1975W An UNKNOWN level of the Service Enhancement parts are now established on *mode cuu*.

Explanation: VMFREC INFO (ALL was used in a situation that prevents VMFREC from knowing the level of the service EXECs and other parts received from files 1 and 2 of the service tape.

System Action: RC = 4.

User Response: Before performing additional service, ensure that the new service EXECs are those you wish to have installed.

Summary of Changes

Technical changes throughout this manual are indicated by vertical bars (|) to the left of the changed material.

Third Edition

Form of Publication: SC23-0364-2

Level of Product: VM/XA System Product Release 2

Date of Publication: March 1989

Changes to this publication:

- **New device support.**

  VM/XA System Product Release 2 fully supports the 3380-K DASD device. The 3380 Starter System can be used with a 3380-K. A new sample directory and minidisk maps for the 3380-K have been added to Appendix C, "VM/XA System Product Starter System Information."

- **Group control system (GCS).**

  GCS is a new component that allows you to implement a native SNA communication network or run RSCS Version 2.

  - A discussion of planning for GCS has been added to Chapter 1, "Introduction."
  - Instructions for installing GCS have been added to Chapter 2, "Installing VM/XA System Product Release 2 with the Starter System (First Level)," and Chapter 3, "Installing VM/XA System Product Release 2 with the Starter System (Second Level)" on page 115.
  - A new chapter, Chapter 16, "Program Update Service or Corrective Service to GCS," shows how to service your GCS system.
  - A new appendix, Appendix E, "Example of Alternate GCS Nucleus Placement," shows how to save your GCS nucleus at a different location than the one in the sample files.

- **CP national language support (NLS).**

  VM/XA System Product Release 2 provides CP HELP files and messages in the following languages:

  - Mixed-case American English (base version)
  - Uppercase American English
  - French
  - German
  - Brazilian Portuguese
  - Japanese (Kanji).

  A discussion of planning for national language support has been added to Chapter 1, "Introduction"; and Chapter 5, "Installing a New System National Language," gives instructions for installing a new system national language.

- **DIRECTXA enhancements.**

  VM/XA System Product Release 2 enhances the function of the DIRECTXA command and the directory build process by providing automatic deactivation of restricted passwords (ADRP). Appendix F, "Restricted Logon Passwords," gives the restricted password list supplied by IBM.

- **Installation using an existing system.**
  - A new chapter, Chapter 4, "Installing VM/XA System Product Release 2 Using an Existing VM/SP or VM/SP HPO System," shows how to install VM/XA System Product Release 2 using an existing VM/SP or VM/SP HPO system.
  - Sample directory entries for the virtual machine used to install VM/XA System Product Release 2 using an existing system appear in Appendix C, "VM/XA System Product Starter System Information."
- **Corrective Service.**

  The procedures for applying corrective service have been brought into line with the procedures for applying program update service. Information on corrective service appears throughout Part 2, "Servicing the System."
- **Serviceability Enhancements.**
  - The following new service EXECs have been developed:
    - VMFOVER EXEC overrides the product parameter file
    - VMFSETUP EXEC establishes the correct minidisk access order for the other service EXECs.

    The new service EXECs are discussed in Appendix B, "EXEC and Command Format Summaries" and throughout Part 2, "Servicing the System."
  - A new appendix, Appendix G, "Controlling Disk String Merges," shows how to perform the disk string merges normally done by the VMFREC EXEC manually and how to prevent VMFREC from merging a string.
- **Servicing CP and CMS.**

  The chapters on servicing CP and CMS (formerly Chapter 9, "Applying Program Update Service to CP," and Chapter 10, "Applying Program Update Service to CMS") have been reorganized to reflect the order of the service process. The new chapters are:

  Chapter 9, "Receiving Program Update Service or Corrective Service for CMS"
  Chapter 10, "Receiving Program Update Service or Corrective Service for CP"
  Chapter 11, "Applying Program Update Service or Corrective Service to CMS"
  Chapter 12, "Applying Program Update Service or Corrective Service to CP"
  Chapter 13, "Rebuilding CMS after Applying Service"
  Chapter 14, "Rebuilding CP after Applying Service."
- **Servicing the Dump Viewing Facility.**

  A new chapter, Chapter 15, "Program Update Service or Corrective Service to the Dump Viewing Facility," shows how to service the dump viewing facility.
- **Local Service.**

  The chapters on applying local service (formerly Chapter 12, "Applying Local Service to CP," and Chapter 13, "Applying Local Service to CMS") have been rewritten. The new chapters are:

  Chapter 18, "Receiving and Applying Local Service"
  Chapter 19, "Emergency Local Service Using the Patch Facility."
- **Removing Service.**

  The chapter on removing service, Chapter 20, "Removing Service from VM/XA SP," has been rewritten.
- **Other service information.**
  - A new appendix, Appendix H, "Service Reference Tables," lists the standard 3-character abbreviations for the various file types used in VM/XA SP and the parts supplied for service.

- A new appendix, Appendix I, "How To Find the PTF Number From the APAR Number," shows the procedure for determining the number of a program temporary fix from the associated APAR number.

- **Duplicate information deleted**.

  The following chapters have been deleted because they duplicate material found in *VM/XA SP Planning and Administration*:

  Chapter 4, "CMS as a Named Saved System"
  Chapter 5, "Licensed Program Planning".

Glossary

A

APAR. Authorized program analysis report.

authorized problem analysis report (APAR). (1) A report of a specific system problem. (2) The code correcting a specific system problem.

automatic software re-IPL. The process by which the control program attempts to restart the system after abnormal termination. This process does not involve the hardware IPL process. See also virtual=real machine recovery.

C

CCS. Console communication services.

CCW. Channel command word.

channel command word (CCW). A doubleword structure that directs an I/O operation on a device or channel and includes pointers to any storage areas associated with the operation. One or more CCWs make up a channel program.

circumventive fix. A fix that branches around the code in error, or avoids executing that code in some other way, rather than correcting the error. A circumventive fix is temporary.

CMS. Conversational monitor system.

control program (CP). The component of VM/XA SP that manages the resources of a single System/370-Extended Architecture system so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of either a System/370 computing system or a System/370-Extended Architecture computing system.

console communication services (CCS). A group of CP routines that interface with the VTAM service machine, providing full VM/XA SP console capabilities for SNA/CCS terminal users.

conversational monitor system (CMS). The component of VM/XA SP that, as a virtual machine operating system, provides interactive time-sharing. CMS allows users to communicate with the system and with each other, to create and edit files, and to develop and run application programs. It operates in either System/370 mode or 370-XA mode under the control of CP.

COR. Corrective service tape.

COR-closed PTF. A PTF that is available on a corrective service tape.

CP. Control program.

corrective service. Service sent to you by IBM to correct a specific problem, supplied on a corrective service tape in the same format as a program update tape.

corrective service tape (COR). A tape containing customized corrective service, sent to you by IBM.

D

DCSS. Discontiguous saved segment.

directory. A CP disk file that includes an entry for each user in the system. The entry defines the characteristics of the user's initial virtual machine configuration. These characteristics include the user ID, the password, normal and maximum allowable virtual storage, virtual device definitions, the privilege class, the dispatching priority, logical line editing characters, and the account number.

discontiguous saved segment (DCSS). A saved segment that occupies one or more architecturally-defined segments. It begins and ends on segment boundaries. It is accessed by its own name. Contrast with member saved segment. See also saved segment, segment, segment space.

disk string. See string.

dump viewing facility. A VM/XA SP component that allows users to display, format, and print data interactively from CP hard and soft abend, stand-alone, and virtual machine dumps, and to process CP trace table data stored on tape or in a system trace file.

dynamic paging area. The area of real storage allocated by CP for V=V machine paging. This

area also contains CP nonresident modules, CP control blocks, CP trace tables, free storage pages, and the alternate processor's prefix storage areas.

E

Expanded Storage. Optional integrated high-speed storage. In VM/XA SP, Expanded Storage may be shared by CP and one or more virtual machines. It may also be dedicated to CP or to a particular virtual machine.

F

full-pack minidisk. A virtual disk that contains all of the addressable cylinders of a real DASD volume.

full-screen mode. In VM/XA SP, the environment in which an entire 3270 display screen is under the control of a program running in a virtual machine.

G

GCS. group control system

group control system (GCS). The component of VM/XA SP that, as a virtual machine supervisor, executes in a group of System/370 virtual machines under CP control to provide an interface that helps support a native Systems Network Architecture (SNA) network.

guest. An operating system running in a virtual machine managed by the VM/XA SP control program. Contrast with host.

guest real storage. The storage that appears real to the operating system running in a virtual machine. Contrast with guest virtual storage, host real storage, and host virtual storage.

guest virtual storage. The storage that appears virtual to the operating system running in a virtual machine. Contrast with guest real storage, host real storage, and host virtual storage.

H

host. The VM/XA SP control program in its capacity as manager of a virtual machine in which another operating system is running. Contrast with guest.

host real storage. The storage that appears real to the control program. If VM/XA SP is running native, this is real storage; if VM/XA SP is running in a virtual machine, this is virtual storage. Contrast with guest real storage, guest virtual storage, and host virtual storage.

host virtual storage. The storage that appears virtual to the control program. Contrast with guest real storage, guest virtual storage, and host real storage.

I

image library. A set of modules, contained in a system data file, that define the spacing, characters, and copy modification data that a 3800 printer uses to print a spool file or that define the spacing and character set that an impact printer uses to print a spool file. See also system data file.

inter-user communication vehicle (IUCV). A generalized CP interface that facilitates the transfer of data among virtual machines.

IUCV. Inter-user communication vehicle.

L

local service. Any service not supplied on a PUT or COR tape. It can be service that you originate, or service that is sent to you by IBM to correct or circumvent a specific problem that you have encountered and reported. Local service can be supplied on a tape, in hard copy, or over the phone.

M

member saved segment. A saved segment that begins and ends on a page boundary. It belongs to from 1 to 64 segment spaces and is accessed either by the segment space name or its own name. Contrast with discontiguous saved segment. See also saved segment, segment, segment space.

message repository file. A type of system data file that contains a set of VM/XA SP messages translated into a national language.

minidisk string. See string.

missing interrupt handler. A CP function for detecting and dealing with real I/O operations that do not complete within a specified time.

multiple preferred guests. A VM/XA SP facility that supports up to six preferred virtual machines when the Processor Resource/Systems Manager™ (PR/SM™) feature is installed in the real machine. See also preferred virtual machine.

N

named saved system (NSS). A copy of an operating system that a user has named and retained in a system data file. The user can load the operating system by its name, which is more efficient than loading it by device number. See also discontiguous saved segment, member saved segment, saved segment, segment space, system data file.

NSS. Named saved system.

O

object-maintained code. Code serviced by replacement of a complete part. See also replacement service, source-maintained code.

P

pageable virtual machine. Synonymous with virtual = virtual machine.

preferred virtual machine. A virtual machine that runs in the V = R area. CP gives this virtual machine preferred treatment in the areas of performance, processor assignment, and I/O interrupt handling. See also multiple preferred guests, virtual = fixed machine, virtual = real area, virtual = real machine.

preventive service. Program update service.

Processor Resource/Systems Manager (PR/SM). A separately orderable feature available with 3090E processors that provides for logical partitioning of the real machine and support of multiple preferred guests. See also multiple preferred guests.

PR/SM. Processor Resource/Systems Manager.

program temporary fix (PTF). A package of one or more APARs.

program update service. Service sent to you on a program update tape, also called preventive service. Not to be confused with update service.

program update tape (PUT). A tape containing customized service. Each service tape contains cumulative service for the customer's products back to the earliest release level of the product still supported.

PTF. Program temporary fix.

PUT. Program update tape.

R

real system operator. Any user who loads and runs VM/XA SP in the real machine. Contrast with virtual machine operator.

remedial service. Service to temporarily correct or circumvent a specific problem that you have encountered and reported, until a PTF becomes available. Remedial service includes corrective service, which IBM supplies on a corrective service tape with the same format as a program update tape, and local service, which IBM supplies in some other format or which you originate yourself.

replacement service. Service by replacement of the entire part that is in error. See also object-maintained code, update service.

S

saved segment. One or more pages of storage that have been named and retained in a system data file. See also discontiguous saved segment, member saved segment, segment, segment space, system data file.

segment. In System/370 architecture, 64 kilobytes of storage. In 370-XA architecture, 1 megabyte of storage. See also saved segment.

segment space. A saved segment composed of up to 64 member saved segments accessed by a single name. A segment space occupies one or more architecturally-defined segments; it begins and ends on segment boundaries. A user with access to a

---

Processor Resource/Systems Manager and PR/SM are trademarks of the International Business Machines Corporation.

segment space has access to all of its members. See also discontiguous saved segment, member saved segment, saved segment, segment.

service tape. Program update tape or corrective service tape.

service virtual machine. A virtual machine that provides system services. These services include accounting, error recording, monitoring, and those provided by supported licensed programs.

SMSG function. A CP function that allows a virtual machine to send a special message to another virtual machine programmed to accept and process the message. See also special message.

SNA. Systems Network Architecture

SNA/CCS terminal. Any terminal accessing VM/XA SP that is managed by a VTAM service machine.

source-maintained code. Code serviced by combining updates with the base code. See also update service, object-maintained code.

special message. A data transmission, made up of instructions or commands, sent from one virtual machine to another via the SMSG function. A special message is processed by the receiving virtual machine and does not appear on the receiver's console. See also SMSG function.

spool file. A collection of data along with CCWs for processing on a unit record device. Contrast with system data file.

string. In applying service, a set of minidisks defined in the product parameter file for a particular use; for example, the BASE2 string, which holds source code, or the DELTA1 string, which holds update files from the program update tape.

SVC 76. In VM/XA SP, a supervisor call instruction that records the error incidents encountered by certain operating systems running in virtual machines. When a virtual machine operating system issues an SVC 76, VM/XA SP translates the virtual storage and I/O device addresses to real addresses, records the information on the VM/XA SP error recording virtual machine, and returns control to the issuing virtual machine. This interface bypasses the virtual machine's own error recording routine, and avoids duplicate error recording.

System/370 mode. A virtual machine operating mode in which System/370 functions are simulated. Contrast with 370-XA mode.

system data file. A collection of data associated with a particular function. Types of system data files include saved segments, NSSs, UCR files, image libraries, message repository files, and system trace files. Because a system data file contains no CCWs, it cannot be processed on a unit record device. Contrast with spool file.

system hold status. A spool file status that prevents a file from being printed, punched, or read until the real system operator releases it. Contrast with user hold status.

system trace file. A type of system data file that contains CP or virtual machine trace data.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration of, networks.

U

UCR file. User class restructure file.

unit record device. A reader, a printer, or a punch.

update service. Service by applying updates to the base code (also called source code) that is in error. Not to be confused with program update service. See also source-maintained code, replacement service.

user class restructure file (UCR file). A type of system data file that contains information used to override the IBM-defined privilege class structure of CP commands, DIAGNOSE instruction codes, and certain CP system functions.

user directory. See directory.

user hold status. A spool file status that prevents a file from being printed, punched, or read until the file owner releases it. Contrast with system hold status.

V

Vector Facility (VF). A hardware feature that provides synchronous instruction processing for high-speed manipulation of fixed-point and floating-point data.

VF. Vector Facility.

V = F machine. Virtual = fixed machine.

virtual = fixed machine (V = F machine). A preferred virtual machine with a fixed, contiguous area of host real storage that does not start at page 0. CP provides performance enhancements for this virtual machine. See also multiple preferred guests, preferred virtual machine, virtual = real area, virtual = real machine, virtual = virtual machine.

virtual machine. In VM/XA SP, a functional equivalent of either a System/370 computing system or a System/370-Extended Architecture computing system. Each virtual machine is controlled by an operating system. VM/XA SP controls the concurrent execution of multiple virtual machines on an actual System/370-Extended Architecture system.

Virtual Machine/Extended Architecture System Product (VM/XA SP). An operating system that allows multiple IBM System/370 and 370-XA operating systems to run simultaneously on a single 370-XA processor. The multiple systems may be used for production, testing, developing application programs, maintenance, and migration. VM/XA SP also provides a high-capacity interactive environment. There are four components: the control program (CP), the conversational monitor system (CMS), the dump viewing facility, and the group control system (GCS).

virtual machine operator. Any user who loads and runs an operating system in a virtual machine. Contrast with real system operator.

virtual = real area (V = R area). A fixed, contiguous section of real storage, starting at page 0, in which preferred virtual machines execute. CP does not page this storage. See also preferred virtual machine, virtual = fixed machine, virtual = real machine.

virtual = real machine (V = R machine). A preferred virtual machine with a fixed, contiguous area of host real storage that starts at page 0. CP provides performance enhancements and an automatic recovery facility for this virtual machine. See also multiple preferred guests, preferred virtual machine, virtual = real area, virtual = real machine recovery, virtual = virtual machine.

virtual = real machine recovery (V = R machine recovery). A CP function that allows the V = R machine to resume operation after most CP abnormal terminations. When possible, the facility reestablishes the V = R machine environment, allowing the operating system running in that virtual machine to perform its own recovery processes. See also automatic software re-IPL.

virtual = virtual machine (V = V machine). A virtual machine that runs in the dynamic paging area. CP pages this virtual machine's guest real storage in and out of host real storage. See also dynamic paging area, virtual = fixed machine, virtual = real machine.

virtual supervisor state. A condition, controlled by a virtual machine's current PSW, during which the control program allows the virtual machine to issue input/output and other privileged instructions. When these instructions are not emulated, the control program intercepts these instructions and simulates their functions for the virtual machine.

virtual wait time. The period during which the control program suspends the processing of a program while a required resource is unavailable.

VM/XA SP. Virtual Machine/Extended Architecture System Product.

VTAM service machine. A collection of networking programs running in a virtual machine that, together with the CP console communication services (CCS) routines, provide full VM/XA SP console capabilities for SNA/CCS terminal users. A VTAM service machine contains either (1) VM/VTAM with VSCS running as an application under control of GCS, or (2) VM/VCNA running as a VTAM application under control of the VSE or VS1 operating system.

V = R area. Virtual = real area.

V = R machine. Virtual = real machine.

V = R machine recovery. Virtual = real machine recovery.

V = V machine. Virtual = virtual machine.

Numerics

370 mode. Synonym for System/370 mode.

370-XA mode. A virtual machine operating mode in which System/370-Extended Architecture functions are simulated. Contrast with System/370 mode.

Bibliography

This bibliography gives the names and order numbers of microfiche and publications about VM/XA System Product.

VM/XA System Product Microfiche

You can order microfiche listings that contain code. The order numbers for the microfiche are:

Order No.	Description
LYC7-0330	VM/XA System Product: CP listings
LYC7-0331	VM/XA System Product: CMS listings
LYC7-0332	VM/XA System Product: GCS listings
LYC7-0334	VM/XA System Product: dump viewing facility listings.

VM/XA System Product Publications

The publications are shown in Figure 65 on page 902. You can order any of them by their individual order numbers or you can order most of them as a group by using a single order number, SBOF-0260. SBOF-0260 provides:

- All unlicensed publications (order numbers that do not begin with LY)
- Enough three-ring binders to hold the publications
- Spine and cover inserts for the binders.

Note that you must order manuals that contain licensed information (manuals with order numbers that begin with LY) through your support personnel. Books that contain licensed information are:

- *VM/XA System Product: Features Summary*, LY27-8058
- *VM/XA System Product: Diagnosis Guide*, LY27-8056
- *VM/XA System Product: CP Diagnosis Reference*, LY27-8054
- *VM/XA System Product: CMS Diagnosis Reference*, LY27-8052
- *VM/XA System Product: Group Control System Diagnosis Reference*, LY27-8060
- *VM/XA System Product: CP Data Areas and Control Blocks*, LY27-8053
- *VM/XA System Product: CMS Data Areas and Control Blocks*, LY27-8051.

As shown in Figure 65 on page 902, VM/XA SP publications are organized into six categories:

1. Evaluation and introduction: information on VM/XA SP concepts.
2. Planning, installation, administration, and service: planning your system and performing system installation and maintenance.
3. Operation and end use: performing system and virtual machine tasks.
4. Application programming: information on using programming interfaces.
5. Diagnosis: information for understanding of VM/XA SP design and to aid in problem diagnosis.
6. Reference: quick retrieving of library usage information, command language syntax, macro instructions, DIAGNOSE codes, and system messages.

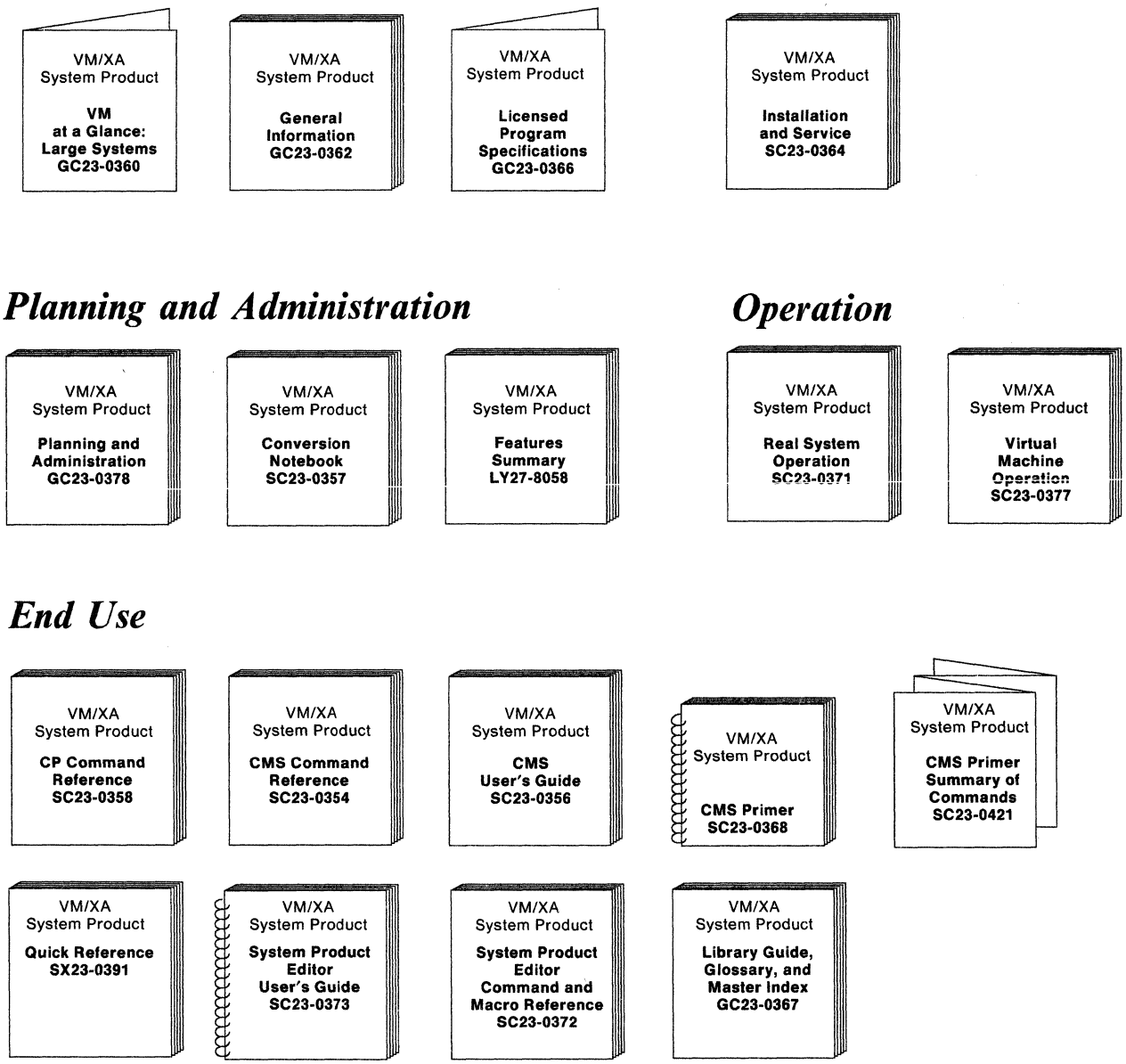


Figure 65 (Part 1 of 2). VM/XA System Product Publications

Evaluation and Introduction: Understanding Basic System Concepts

The evaluation and introduction publications for VM/XA SP are:

- *VM/XA System Product: Licensed Program Specifications*, GC23-0366

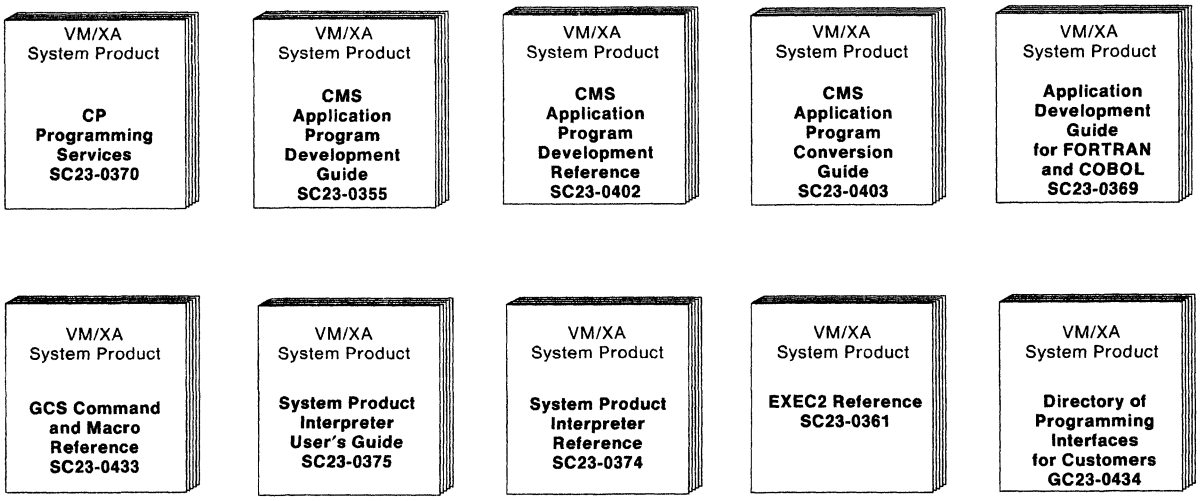
  Provides information on the warranted functions of VM/XA SP and describes the specified operating environment.

- *VM at a Glance: Large Systems*, GC23-0360

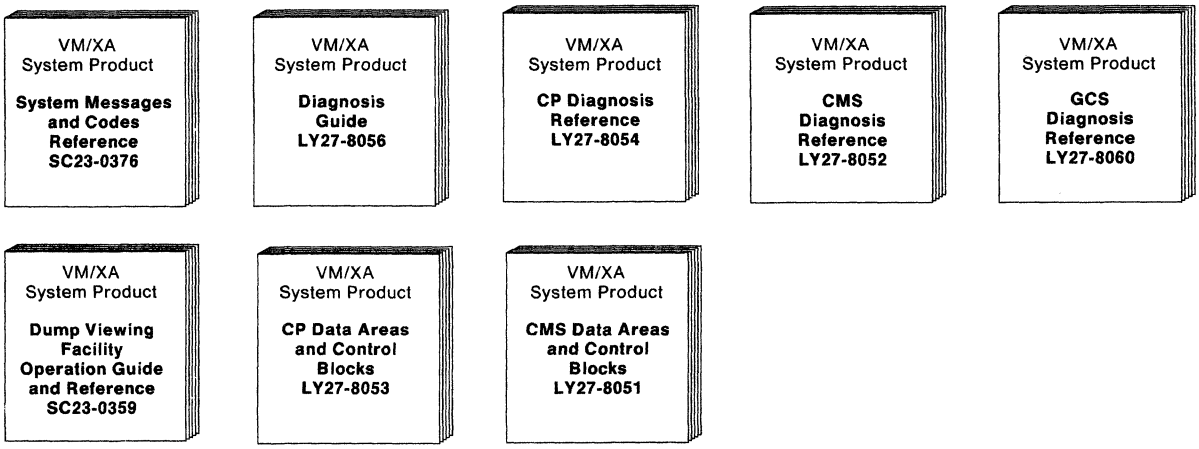
  Presents an overview of the features of each of the large VM systems: the VM/XA Systems Facility, VM/SP High Performance Option, and VM/XA System Product.

- *VM/XA System Product: General Information*, GC23-0362

Application Programming



Diagnosis



Binders and Inserts

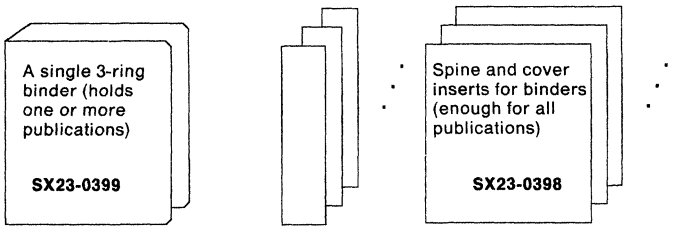


Figure 65 (Part 2 of 2). VM/XA System Product Publications

Provides general and planning information for VM/XA SP. It can help you decide whether VM/XA SP can fill your needs.

- *VM/XA System Product: Conversion Notebook*, SC23-0357

  Provides migration and compatibility information for customers migrating from VM/SP HPO Release 5 and VM/XA SF Release 2.

Planning, Installation, Service, and Administration: Generating and Maintaining the System

- *VM/XA System Product: Planning and Administration*, GC23-0378

  Presents system planning concepts for VM/XA SP and virtual machine planning concepts for guest operating systems. Planning topics include suggestions for defining your real system configuration and building and updating your directory. This book discusses running these operating systems under VM/XA SP: MVS/SP, MVS/XA™, VSE/SP, VM/SP, and VM/SP HPO.

  This book also provides information on how to manage your system. Administration topics include:

  - Setting up virtual machines for accounting, error recording, and CMS batch
  - Setting up the programmable operator
  - Redefining command privilege classes
  - Defining and managing saved segments and named saved systems
  - Tuning the system
  - Reference information on the VM/XA SP monitor.

- *VM/XA System Product: Installation and Service*, SC23-0364

  Gives step-by-step procedures for generating VM/XA SP and describes how to apply service updates to your system.

Operations and End Use: Making the System Work for You

- *VM/XA System Product: Real System Operation*, SC23-0371

  Provides a task-oriented source for real system operations. In step-by-step format it describes the procedures and commands used to perform each real system task.

- *VM/XA System Product: Virtual Machine Operation*, SC23-0377

  Provides a task-oriented source for virtual machine operations. In step-by-step format it describes the procedures and commands used to perform each virtual machine task.

- *VM/XA System Product: CMS User's Guide*, SC23-0356

  Provides information on using CMS.

- *VM/XA System Product: CMS Primer*, SC23-0368

  Provides a tutorial approach to learning CMS.

- *VM/XA System Product: System Product Editor User's Guide*, SC23-0373

  Provides information about using the System Product Editor.

Application Programming: Using Programming Interfaces

- *VM/XA System Product: CP Programming Services*, SC23-0370

  Provides reference and usage information for the following CP services and macros:

  - The DIAGNOSE codes
  - The IUCV macro
  - CP system services.

MVS/XA is a trademark of the International Business Machines Corporation.

- *VM/XA System Product: CMS Application Program Development Guide*, SC23-0355

  Helps you use the assembler language macros and functions of CMS in your assembler language application programs. It describes how to manage storage, perform I/O, handle interrupts, process abends, load and start programs, and exploit 31-bit addressing. It also includes message repository information.

- *VM/XA System Product: CMS Application Program Conversion Guide*, SC23-0403

  Helps you convert your existing CMS assembler language application programs so that they run on the CMS provided with VM/XA SP. It summarizes the differences between the CMS provided with VM/XA SP and previous versions of CMS, it describes the tasks you may need to perform in converting your programs, and it points you towards other books that can help you convert your programs.

- *VM/XA System Product: Application Development Guide for FORTRAN and COBOL*, SC23-0369

  Provides information on how to use the CMS environment to develop and execute FORTRAN and COBOL application programs. The book contains such information as:

  - How to use the System Product Editor to create an application program
  - How to load, compile, and execute selected supported licensed programs.

- *VM/XA System Product: System Product Interpreter User's Guide*, SC23-0375

  Provides information about using the System Product Interpreter.

- *VM/XA System Product: Directory of Programming Interfaces for Customers*, GC23-0434

  Helps you locate application programming interface information in the VM/XA SP library.

Diagnosis: Understanding System Design

- *VM/XA System Product: Diagnosis Guide*, LY27-8056

  Provides diagnostic information. It describes how to locate problems within the VM/XA SP control program, and how to describe and report problems to IBM support personnel. The diagnosis reference publications describe how the system works. You should use them as supplements to this book.

- *VM/XA System Product: CP Diagnosis Reference*, LY27-8054

  Describes each of the major VM/XA SP control program facilities. Also contains a module cross-reference list.

- *VM/XA System Product: CMS Diagnosis Reference*, LY27-8052

  Describes each of the major conversational monitor system facilities.

- *VM/XA System Product: Group Control System Diagnosis Reference*, LY27-8060

  Describes step-by-step procedures for identifying problems in each of the major facilities of the group control system and lists important GCS control blocks.

- *VM/XA System Product: CP Data Areas and Control Blocks*, LY27-8053

  Lists the data areas and control blocks used by the VM/XA SP control program.

- *VM/XA System Product: CMS Data Areas and Control Blocks*, LY27-8051

  Lists the data areas and control blocks used by CMS.

- *VM/XA System Product: Dump Viewing Facility Operation Guide and Reference*, SC23-0359

  Describes step-by-step procedures for using the dump viewing facility. The publication is also a reference for dump viewing facility commands and messages.

Reference: Retrieving Information Quickly

- *VM/XA System Product: CP Command Reference*, SC23-0358

  Provides complete descriptions of the commands used to communicate with VM/XA SP, including usage notes. The commands are in alphabetical order.

- *VM/XA System Product: CMS Command Reference*, SC23-0354

  Provides complete descriptions of the commands used to communicate with the CMS component of VM/XA SP. The commands are in alphabetical order.

- *VM/XA System Product: Group Control System Command and Macro Reference*, SC23-0433

  Provides complete descriptions of the commands used to communicate with the GCS component of VM/XA SP and the macros that can be used in GCS application programs. This book also includes information about writing REXX EXECs for use under GCS.

- *VM/XA System Product: Features Summary*, LY27-8058

  Provides a comprehensive survey of VM/XA SP at a higher level than the *VM/XA SP CP Diagnosis Reference*. Topics cover such areas as:

  - Supported features, hardware, and operating systems
  - CP-owned direct access storage, CP virtual storage, and real storage organization
  - CP virtual and real machine management
  - CMS, GCS, and the dump viewing facility
  - VM/XA SP performance facilities and installation.

- *VM/XA System Product: System Messages and Codes Reference*, SC23-0376

  Contains the system messages generated by the VM/XA SP CP, CMS, and GCS components. For each message, the publication provides:

  - The message number
  - The message text
  - An explanation of why the message was issued
  - System action
  - Recommended operator action (if any)
  - Recommended user action (if any)
  - Return code (if any).

  The publication also documents all abend codes and wait state codes, as well as the reason for each code and the recommended action.

- *VM/XA System Product: CMS Application Program Development Reference*, SC23-0402

  Describes the CMS programming interface. It includes descriptions of the CMS macros, DOS macros, and external-use control blocks.

- *VM/XA System Product: Quick Reference*, SX23-0391

  Shows only the command syntax of all the VM/XA SP commands. The commands summarized in this publication are described in detail in the *VM/XA System Product: CP Command Reference*, the *VM/XA System Product: CMS Command Reference*, the *VM/XA System Product: Dump Viewing Facility Operation Guide and Reference*, and the *VM/XA System Product: Group Control System Command and Macro Reference*.

- *VM/XA System Product: System Product Editor Command and Macro Reference*, SC23-0372

  Describes the system product editor commands, in alphabetical order.

- *VM/XA System Product: System Product Interpreter Reference*, SC23-0374

  Describes the system product interpreter statements, in alphabetical order.

- *VM/XA SP EXEC 2 Reference*, SC23-0361

  Describes the EXEC 2 control words, in alphabetical order.

- *VM/XA System Product: Library Guide, Glossary, and Master Index*, GC23-0367

  Provides an overview of the library's structure, a glossary, and a means for directly locating specific information within a manual or manuals.

- *VM/XA System Product: CMS Primer Summary of Commands*, SC23-0421

  Contains summary information about commands described in the *VM/XA System Product: CMS Primer*.

Index

A

B

C

D

E

F

M

N

O

P

R

S

T

U

V

W

X

Z

Numerics

Special Characters

Virtual Machine/Extended Architecture
System Product
Release 2

SYSTEM USABILITY COMMENTS

Please use this form to communicate your comments about the usability of the VM system, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the Product Usability Department for appropriate review and action, if any. Comments may be written in your own language; English is not required.

System Information

If you answer **No**, please explain.

	Yes	No
• Does the VM system meet your needs?	☐	☐
• Is it easy to use and understand?	☐	☐
• Are the commands/messages easy to understand and use?	☐	☐
• Are the HELP facilities appropriate?	☐	☐

Customer Information

- What is your occupation? _____
- How long have you been in this occupation? _____
- How long have you been using VM? _____
- Indicate the tasks your job involves:

Evaluation	☐	Planning	☐
Installation	☐	Administration	☐
Customization	☐	Operations	☐
Diagnosis	☐	End Use	☐
Other	☐		

Your Comments:

We appreciate your comments.

If you would like a reply, please supply your name and address on the reverse side of this form. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automatic mail-sorting equipment. Please use pressure-sensitive or other gummed tape to seal this form.

System Usability Comments

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 49MA MS 925
Neighborhood Road
Kingston, New York 12401

Fold and Tape | Please Do Not Staple | Fold and Tape

If you would like a reply, *please print:*

Your Name _____

Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____

IBM®

PRINTED IN U.S.A.

Sytem Product
Release 2

READER'S COMMENT FORM

Serviceabilty Enhancements
Program Update Information
APAR VM 37518

Order No. GC23-0503-0

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

How did you use this publication?

[ ] As an introduction
[ ] As a reference manual
[ ] For another purpose (explain) _____

[ ] As a text (student)
[ ] As a text (instructor)

_____

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

What is your occupation? _____

Newsletter number of latest Technical Newsletter (if any) concerning this publication: _____

If you wish a reply, give your name and address: _____

_____

_____

IBM branch office serving you _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automatic mail-sorting equipment. Please use pressure-sensitive or other gummed tape to seal this form.

GC23-0503-0

Reader's Comment Form

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 52Q MS 511
Neighborhood Road
Kingston, New York 12401

Fold and Tape | Please Do Not Staple | Fold and Tape

IBM®

PRINTED IN U.S.A.