VM/ESA Release 2.2 Overview and Usage Experiences

Document Number GG24-4219-00

June 1994

International Technical Support Organization
Poughkeepsie

Take Note!

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

First Edition (June 1994)

This edition applies to Virtual Machine/Enterprise Systems Architecture (VM/ESA), program number 5684-112.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. H52 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

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

Abstract

This document provides a single source insight into the new function and enhanced features of VM/ESA Release 2.2. The discussions within include usage experiences, recommendations, and easy to follow examples.

This document was written for technical professionals wishing to increase their knowledge of this topic. You may consider reading this publication if you plan on marketing, using, installing, migrating to, or purchasing VM/ESA Release 2.2. Some knowledge of prior VM/ESA releases is assumed.

VM (216 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help customers and system engineers understand, install, and use the features of VM/ESA Release 2.2. The information is not intended as a specification of any programming interfaces that are provided by VM/ESA Release 2.2. However, we intend it to complement the product documentation listed in the Publications section of the IBM Programming Announcement for VM/ESA Release 2.2.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT 06904 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	APL2
BookMaster	C/370
CICS	CICS/ESA
DB2	ECFORMS
ES/9000	ESA/370
ESA/390	GDDM
IBM	MVS/ESA
OfficeVision/VM	OS/2
Print Services Facility	PSF
QMF	RACF
S/370	S/390
SAA	SQL/DS
System/370	System/390
Virtual Machine/Enterprise Systems Architecture	Virtual Machine/Extended Architecture
VM/ESA	VM/XA
VSE/ESA	VTAM
3090	

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

Advantis	Advantis
DEC	Digital Equipment Corporation
HP	Hewlett-Packard Company
Open Software Foundation	Open Software Foundation, Inc.
OSF	Open Software Foundation, Inc.
Motif	Open Software Foundation, Inc.
Novell Netware	Novell Corporation
Sun Microsystems	Sun Microsystems, Inc.
UNIX	Unix System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc.
Windows	Microsoft Corporation

Other trademarks are trademarks of their respective companies.

Preface

With the introduction of VM/ESA* (Virtual Machine/Enterprise Systems Architecture*) Release 2.2, IBM has continued the evolution and enhancement of its valued mid-range and large system operating system.

The main focus of this new release is on the VM/ESA investment areas of cost of computing, and core business; however, there are significant enhancements in the VM/VSE synergy, and Open and Client/Server computing areas as well.

This document contains an overview of all the new features and enhancements for VM/ESA Release 2.2, along with usage experiences, examples, and recommendations.

Who Should Read This Document

This document is intended for IBM technical professionals and customer personnel seeking information about VM/ESA Release 2.2.

Readers include IBM and customer technical professionals and, in general, VM systems specialists, systems programmers, and system planners.

How This Document is Organized

The document is organized as follows:

- **Chapter 1, "VM/ESA Introduction and Release 2.2 Overview"**

  This chapter contains a management overview of VM and is packed with facts and details about the product, including an introduction of what is new in this release.

- **Chapter 2, "CP Enhancements"**

  In this chapter, CP enhancements for the spool system, the scheduler, and connectivity are addressed. There are many other important enhancements worth reading here.

- **Chapter 3, "CMS Enhancements"**

  This chapter contains not only the enhancements to CMS, but also introduces a new integrated interface between system support personnel, products or applications, and the users: VMLINK.

- **Chapter 4, "VMSES/E Enhancements"**

  In this chapter, all the major VMSES/E enhancements are discussed. Your system is even easier to service now.

- **Chapter 5, "Shared File System Enhancements"**

  You will want to read this chapter if your users are fond of the Shared File System. SFS now has a better backup interface, administrator enhancements, and a new general user command.

- **Chapter 6, "Group Control System (GCS) Enhancements"**

  GCS is getting ready for the connectivity future. This chapter tells you how.

- **Chapter 7, “Problem Determination Aids”**

  A new non-disruptive dump and some tools once used by the VM masters themselves are introduced in this chapter. For debuggers, there is a TRSOURCE section worth reading.

- **Chapter 8, “Hardware Support”**

  In this chapter, you can read all about VM′s latest hardware support, including CPUs and DASD.

- **Appendix A, “VM Open System Statement of Direction”**

  VM is heading into the open enterprise. This section tells you what is planned. A reprint of the current IBM statements of direction for VM is included.

- **Appendix B, “Local Modification to SYSPROF EXEC”**

  With VMSES/E, updating your SYSPROF EXEC is a little tricky the first time. This chapter shows how simple it can be.

- **Appendix C, “VM/ESA Release 2.2 IPL Using Integrated Console Facility”**

  In this section, we show you how to use the integrated hardware console.

- **Appendix D, “QUIESCE/RESUME Request with a 3990 Model 6”**

  This section explains how the new 3990 Model 6 reacts to a control unit initiated quiesce and resume.

- **Appendix E, “Monitor and Accounting Record Changes”**

  This section lists the changes to the monitor and accounting records.

- **Appendix F, “VMLINK Setup, Exit Activation and Using Examples”**

  In this section, we expose all the details and examples pertaining to VMLINK.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *VM/ESA Running Guest Operating Systems*, SC24-5522
- *VM/ESA CMS Pipelines Reference*, SC24-5592
- *VM/ESA Planning and Administration*, SC24-5521
- *VM/ESA: REXX/EXEC Migration Tool*, GC24-5607
- *VM/ESA: Conversion Guide and Notebook for VM/SP, VM/SP HPO, VM/ESA (370 Feature)*, SC24-5654
- *VM/ESA: Conversion Guide and Notebook for VM/XA SP and VM/ESA*, SC24-5525
- *VM/ESA CP Command and Utility Reference*, SC24-5519
- *VM/ESA CP Programming Services*, SC24-5520
- *VM/ESA CMS Command Reference*, SC24-5461
- *VM/ESA Release 2.2 Performance Report*, GC24-5673-01
- *VM/ESA Connectivity Planning, Administration and Operation*, SC24-5448
- *Washington Systems Center FLASH 9313*

- *VM/ESA CMS Application Development Reference for Assembler*, SC24-5453
- *VM/ESA CMS Pipelines User's Guide*, SC24-5609
- *VM/ESA CMS Pipelines Tutorial*, GG66-3158
- *VM/ESA REXX User's Guide*, SC24-5465
- *VM/ESA REXX Reference*, SC24-5466
- *3270 Information Display: Data Stream Programmer's Reference*, GA23-0059
- *VM/ESA CMS Application Development Reference*, SC24-5451
- *VM/ESA VMSES/E Introduction and Reference*, SC24-5444
- *VM/ESA Service Guide*, SC24-5527
- *ITSO VMSES/E Primer: Concepts and Experiences*, GG24-3851-02
- *VM/ESA SFS and CRR Planning, Administration and Operation*, SC24-5649
- *VM/ESA Group Control System*, SC24-5531
- *VM/ESA Dump Viewing Facility*, SC24-5530
- *S/390 Rainbow Books Collection*, SK2T-2177

In addition, the following publications are available through IBM Service using their APAR or PTF number:

- *VM/ESA Installation Guide*, (UM98122)
- *Informational APAR II06583*
- *VM/ESA APAR VM51012*

International Technical Support Organization Publications

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

To get listings of redbooks online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order Redbooks

IBM employees may order redbooks and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their IBM branch office.

You may order individual books, CD-ROM collections, or customized sets, called GBOFs, which relate to specific functions of interest to you.

Acknowledgments

This publication is the result of a residency conducted at the International Technical Support Organization, Poughkeepsie.

The residency was coordinated by:

Scott Vetter IBM ITSO Poughkeepsie

The authors of this document are:

Timothy Maguire IBM Australia

Kieran Organ IBM United Kingdom

Veronica Quatrocchi IBM Italy

Martin Strohmaier IBM Germany

We would also like to acknowledge the professionals who took time to review this document, and provided invaluable advice and guidance during its development:

Marci Beach IBM Endicott

Gary Beck IBM Endicott

William Bitner IBM Endicott

Kay Blake IBM Endicott

Robert Boos IBM Endicott

Ron Bos IBM The Netherlands

Currie Boyle IBM Canada

Frank Brice IBM Endicott

Larry Brocious IBM Endicott

Kris Buelens IBM Belgium

Melissa Carlson IBM Endicott

Andrew Coleman IBM Endicott

Guy De Ceulaer IBM Belgium

Bill Degli-Angeli IBM Endicott

Len Delmolino IBM Endicott

Jim Elliott IBM Canada

Susan Farrell IBM Endicott

Scott Fegan IBM Endicott

Glenda Ford IBM Endicott

John Franciscovich IBM Endicott

Annette Fuller IBM Endicott

Francisco Grossi IBM Brazil

John Harris IBM Endicott

Nasrin Housh IBM Endicott

Rhonda Kuiper	IBM Endicott
Mark Lorenc	IBM Endicott
Jim McCormick	IBM Endicott
Betty McGlynn	IBM Endicott
Tom Myers	IBM Endicott
Scott Nettleship	IBM Endicott
Agustin Palacio	IBM Endicott
Sandra Pickering	IBM Endicott
Carol Rozella	IBM Endicott
Jon Ruhl	IBM Endicott
Leo Schumacher	IBM Endicott
Carla Sirgany	IBM Endicott
Theeraphong Thitayanun	IBM Thailand
Susan Timashenka	IBM Endicott
Tom Vail	IBM Endicott
Mary Ellen Vendryes	IBM Endicott
William Wandell	IBM Endicott
Nina Watrous	IBM Endicott
Steve Wilkins	IBM Endicott
Linda Wolff	IBM Endicott
Terrence Woodnorth	IBM Endicott
Mark Zielenski	IBM Endicott

This document was developed entirely under VM/ESA, using the CMS Shared File System. It was formatted using IBM BookMaster* Release 4.0 (program number 5688-015) and IBM Document Composition Facility (program number 5748-XX9). It was printed on an IBM 3827 printer using IBM Print Services Facility*/VM (program number 5664-198).

Chapter 1. VM/ESA Introduction and Release 2.2 Overview

The function of computing within an enterprise has changed over the past twenty years. Initially, the Information Technology (IT) department helped automate the business support activities. Now, its primary function is to provide the business with timely information that is used to gain competitive advantage in the market place. This drive for competitive edge means that the demand for new functions to exploit the latest technology grows.

If we look at the financial aspects of running an IT department, we find that the enterprise wants a higher return from its investment in IT infrastructure. However, this must be achieved with frozen resource and stringent justifications required for an increase in IT spending.

One of the numerous challenges facing the IT department is to balance the skilled resources within the department between maintaining system software and maximizing the benefit from installed applications. This must be achieved with minimum impact on system availability, at a fixed cost and within project deadlines.

1.1 The VM/ESA Strategy

IBM*, like other companies, strives to reduce its product cycles to meet customer demands. At the same time, the quality of the product must be maintained. In response to customer requirements, IBM is increasing the rate at which new functions are added to the VM/ESA* operating system. Over the past four years, we have seen five releases of the VM/ESA operating system. Each has provided extra functions requested by our customers, and also, the quality of code shipped with each release continues to improve.

Figure 1 shows the number of outages, measured by the Early Support Program (ESP), for the various releases of VM. This shows the quality of the delivered code has improved from release to release.

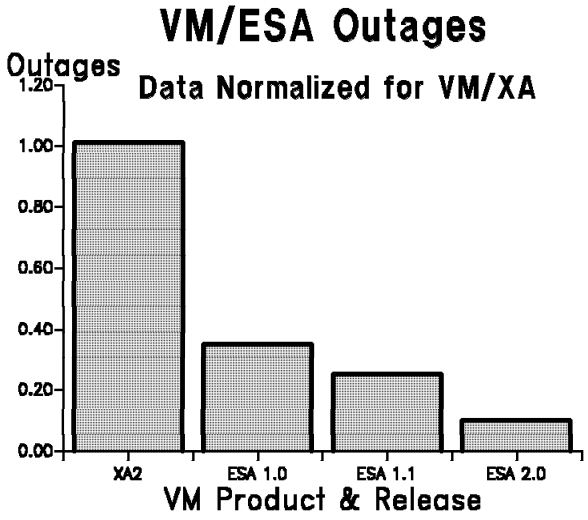


Figure 1. VM Outages Graph

1.2 Why Choose VM/ESA?

Some of the reasons for choosing VM/ESA for computing in the 1990s include:

- Guest Operating System Support

  VM/ESA is IBM′s premier operating system for supplying multiple machine images. VM provides a platform not only for hosting the traditional VSE and MVS operating systems, but also for dependent guests, such as MUMPS/VM and AIX*/ESA.

  The complete list of guest operating systems supported by VM/ESA is available in *VM/ESA Running Guest Operating Systems*.

- Interactive Computing

  VM has always provided an ideal base for interactive computing. VM/ESA is no exception to this. It provides the facilities for running applications, such as office (OfficeVision*/VM), publishing (DCF, BookMaster, BrowseMaster), Computer Aided Design, and numerous other applications. VM/ESA is a true interactive application platform and provides cross platform application environment support. In addition, VM/ESA complements these services with VM-specific Application Programming Interface (APIs), such as the Callable Services Library and VM Data Spaces.

- Application development.

  VM/ESA is an ideal base for applications development. It provides the facilities for coding (integrated editing using XEDIT), compilation (using numerous compilers supported under VM/ESA), and running applications.

- Server Based Computing

  This type of data processing involves the services and resources of VM/ESA to supply request/reply communication based applications. VM/ESA provides extended storage, connectivity, processing capacity and other services to directly attached and LAN attached workstations and virtual machines.

1.3 VM/ESA Investment Areas

IBM has focused its investment in VM/ESA into four major areas:

- Cost of Computing

  Reduce the Cost of Computing to the customer by:

  – Using and supporting lower cost System/390* microprocessor technologies
  – Improving inter-operation with workstations
  – Improving the quality of the VM system to minimize cost of maintaining this environment
  – Improving system management tools to minimize administration and operation costs
  – Providing tools to aid productivity of programmers as they develop and maintain VM applications.

- Core Business

  Protect and enhance the customer's investment in VM. This includes enhancements to the traditional areas of:

  - Office
  - Guest support
  - Database
  - Application development

- VM/VSE Synergy

  Improve the interfaces and communications between VM and VSE running as a guest operating system. This is one of the largest uses of VM and as a result was highlighted for special attention to ensure that sufficient resource is applied to this key area of customer value.

- Open and Client/Server Computing

  Extend VM's, already wide, support of Open standards. This enables customers to achieve the benefits of distributed and PC technologies while gaining maximum value from their VM system.

This chapter describes the enhancements made in each of these four key areas in VM/ESA Release 2.2, and how they can benefit your organization. Later chapters discuss the technical details of these enhancements.

1.3.1 Cost of Computing

VM/ESA Release 2.2 introduces many new enhancements that help minimize the overall cost of computing. A brief overview of these enhancements is given in this section.

Spool File Back-up Enhancements

A new command, SPXTAPE, has been added to improve spool file backup and recovery of large volumes of spool files. SPXTAPE saves and loads spool and system data files to and from tape. These files range from E-mail to printer image library files or system saved segments.

SPXTAPE has greatly improved performance characteristics relative to the current SPTAPE command. In our tests on a second-level system with 8,000 spool files on one spool volume, we saw a three-fold reduction in elapsed time to dump to tape all the spool files. Laboratory measurements have shown a ten-fold reduction in elapsed time and a 30% reduction in tapes required on a fully configured system. Much larger reductions in tape requirements are possible for customers with many small spool files.

This improved performance makes it practical to backup spool files on a regular basis, improving your ability to guard against data loss.

Refer to section 2.2, "SPOOL Backup Enhancements - SPXTAPE" on page 26 for further details on SPXTAPE.

System Maintenance Improvements

Service process improvements continue to make it easier to maintain and update VM/ESA. With VMSES/E on Release 2.2, the system′s programmer can create a private version of any object he has serviced or updated with a local modification. This allows for testing of the updated part before committing the update to the software inventory.

In addition, the Product Service Upgrade (PSU) procedure for applying Recommended Service Upgrade tapes (RSUs) is enhanced with additional automation in the area of RSU planning. This automation includes identification of local modifications which need to be reworked and helps in eliminating a manual, time consuming, and error prone process.

Refer to Chapter 4, “VMSES/E Enhancements” on page 91 for further details on these two enhancements and other improvements that have been made to the VMSES/E component of VM/ESA Release 2.2

Improved Hardware Support

VM/ESA Release 2.2 supports the new ES/9000* processors, including the ES/9000 9221 models. These expand the capabilities of the air-cooled rack-mounted family and functional capabilities of the 9221 processors by using new technology and processor design.

In addition, VM/ESA Release 2.2 supports the new ES/9000 Model 9021-9X2 in basic mode. This 10-processor machine offers a high-end growth option to VM customers exceeding the capacity of their existing processor.

VM/ESA Release 2.2 supports the Controlled Unit Initiated Reconfiguration (CUIR) feature of the 3990 Model 6. The automatic control provided by CUIR greatly simplifies the actions required by operations personnel to manage 3990 service requirements, and assists in the avoidance of errors caused by the manual actions in use today.

Also, support has been provided to allow the operator console of your ES/9000 mainframe to be used as a VM system console.

Refer to Chapter 8, “Hardware Support” on page 135 for further details on the new enhancements made to VM/ESA Release 2.2 hardware support.

System Availability Improvements

Several improvements have been made to VM/ESA Release 2.2 to minimize the amount of time a system is unavailable due to maintenance application or problem determination.

Customers who have a problem with their system now can obtain a full CP dump without a re-IPL. This new capability to dump the entire system is similar to a hard abend dump. However, the system does not need to be re-IPLed. After the dump has been taken, users can resume their operation with no loss to temporary disks or virtual machines, including V=R and V=F guest machines.

Also, enhancements to the system′s data collection facility (TRSOURCE) allows greater control over the information collected through a trace. This ensures that the correct information is saved and system degradation due to the data collection is minimized.

For further details on these two enhanced features, refer to Chapter 7, “Problem Determination Aids” on page 125.

Dynamic time zone changes are now supported by CMS, so it is no longer necessary to re-IPL CMS to get correct file time stamps after a time zone change (biannual or other).

CMS logical segment support no longer requires the SYSTEM SEGID file to be copied to the S-disk when only logical segment addresses are changed, thus reducing the number of times CMS must be re-saved and re-IPLed.

For further details on these two enhanced features, refer to Chapter 3, “CMS Enhancements” on page 71.

1.3.2 Core Business

VM/ESA Release 2.2 has also been enhanced in the area of core business. These enhancements aid user productivity while users are performing their day to day functions.

Share Capping

VM/ESA Release 2.2 has improved the algorithm the scheduler uses to distribute CPU resource to users. A maximum limit can be placed on the users share of the CPU to limit CPU consumption and prevent users from monopolizing the system.

In previous releases of VM/ESA, a minimum share could be set to control user IDs. However, if surplus CPU resource was available, the scheduler did not have an algorithm for distribution of this resource. Release 2.2 has been enhanced to provide an algorithm that allows surplus CPU resource to be distributed to users in proportion to the user′s specified share.

Refer to section 2.3, “Share Capping and Proportional Distribution” on page 41 for further details on share capping.

Improved Access to Applications

A tool to link disks and directories has been added in VM/ESA Release 2.2. VMLINK helps both the general user and the VM system support personnel. It is a menu-driven, common access tool and an Application Program Interface (API). It allows end users to easily determine what applications are available on the system and, using nicknames, it carries out the necessary links and accesses to disks and Shared File System directories under the covers.

If information changes, a single update can be made in the NAMES file(s). Users no longer need to remember the access commands. Calls to support personnel are reduced.

Refer to 3.1, “VMLINK” on page 71 for further details on the new VMLINK tool.

Shared User ID Support

A new Logon BY function has been added to VM/ESA Release 2.2. VM/ESA now allows up to 8 users to share log on to a user ID using their own user ID′s password. This enhances the security and audit ability of privileged user IDs, such as MAINT and OPERATOR, that naturally tend to have multiple users.

Refer to section 2.6, “Logon BY” on page 58 for further details on the new Logon BY enhancements.

CMS Pipelines Improvements

In VM/ESA Release 2.2, several new stage commands have been added to CMS Pipelines. Many of the existing commands have also been improved to provide additional function. These new and enhanced commands provide improvements in record selection, running host commands, string handling, and creating user-written stages. For details of these enhancements, refer to section 3.2, “CMS Pipelines Enhancements” on page 76.

For users who do not have access to *VM/ESA CMS Pipelines Reference*, Help for Pipelines has been enhanced to include Related Help. Related Help makes it easier to view the help files of commands that are related to the stage command you are viewing.

Guest Support

VM/ESA continues to invest in new architecture support. Release 2.2 provides the preferred guest support for using the Asynchronous Data Mover Facility (ADMF) in ES/9000 processors, as announced in a May 1993 Statement of Direction.

ADMF is a new hardware feature used to move data between expanded and main storage asynchronously from the CPU. This allows subsystems, such as DB2* on MVS guests, to off-load the data movement from the CPU, placing the demand on the channel subsystem, providing the potential for improved internal throughput.

VM/ESA Release 2.2 supports the ESA/390* Subspace Group Facility for guest use, as announced in a February 1993 ES/9000 processors statement of direction.

Refer to Chapter 8, “Hardware Support” on page 135 for further information on these two topics.

Support of the Cross-System Coupling Facility (XCF), added in Release 2.2, allows MVS guests to operate as a sysplex, provided they are all located second level under the same VM system. New commands and controls enable this function. This function should not be confused with the Sysplex Timer and the Coupling Facility, which are currently not supported on VM or VM guest system.

Refer to section 2.8, “Working Allegiance (MVS Sysplex DASD Sharing)” on page 64 for further information.

1.3.3 VM/VSE Synergy

VM and VSE′s working relationship continually improves. The enhancements are summarized in the following sections.

Minidisk Caching Enhancements

In VM/ESA Release 2.2 minidisk caching has been upgraded to provide caching of a full track of DASD in main storage, expanded storage or a combination of both. The flexibility is added to use and set the size of main storage as well as expanded storage portions of the cache. New commands have been added to specify the devices to be cached. Any VM guest can now cache its data. This feature is also available to all CMS minidisks and guest system regardless of the disk block size.

Guest systems can do more work in less time with the ability to cache their data. Prior to VM/ESA Release 2.2, only CMS users could receive the benefits from minidisk caching. Now, performance improvements benefit both guests and CMS users.

Laboratory measurements using VSE guest workloads have shown response time and batch elapsed time reductions of up to 50% and processor capacity improvements of up to 7% with minidisk caching. Minidisk caching uses main storage, expanded storage, or both to reduce DASD read I/Os. Accordingly, the degree of performance benefit depends on the extent of DASD read activity in the workload and the amount of storage that is made available.

In applicable cases, minidisk caching is used with virtual disks in storage to achieve even greater improvements. Laboratory measurements, using a VSE guest workload, have shown elapsed time reductions of up to 60% and processor capacity improvements of up to 10% through the combined use of minidisk caching and virtual disks in storage.

Refer to section 2.1, “Minidisk Caching Enhancement” on page 17 for further information.

PVM Console Services for VSE

VM/Pass-Through Facility Version 2 Release 1.1, is enhanced with APAR VM57538 to provide a function that will enable VSE guest users access to a VM session through VSE/VTAM. VTAM* is not required on the VM system.

The APAR provides a VTAM application that is installed on your VSE guest. This application connects to the PVM V2 server on the VM host system. This support is limited to terminal access only. It does not provide support for LU 6.2 communication.

1.3.4 Open and Client Server Computing

IBM has made several Statements of Direction regarding VM/ESA′s support of POSIX and DCE standards, details of these statements of direction are provided in Appendix A, “VM Open System Statement of Direction” on page 143.

In addition to these statements of direction, several enhancements have been made to VM/ESA Release 2.2 in the area of networking and connectivity.

Connectivity Enhancements

With VM/ESA Release 2.2, those customers who use Inter-System Facility for Communications (ISFC) in communication services (CS) collections no longer need to be fully interconnected. Applications within a CS collection can communicate with each other without having a direct connection between the source and target systems. ISFC routes the communication through the intermediate systems. This applies to APPC and to the new IUCV support through ISFC.

Users of Inter-User Communication Vehicle (IUCV) have improved flexibility as well. Servers that use IUCV for communications no longer have to reside on the same system as the requesting virtual machines. IUCV, along with ISFC, transparently locates the target server within a CS collection and routes the connection to that target.

Through intermediate node routing, elimination of the “star network” reduces the number of channel-to-channel (CTC) adapters needed to support a CS connection, thereby reducing cost.

Servers and other virtual machines that use IUCV or APPC for communications benefit from improved IUCV/APPC performance. A multiple processor (MP) capable server now is able to use IUCV and APPC from multiple virtual processors on a single virtual machine. CPU utilization may be improved, more clients can be supported by the server, and server performance may be improved once this feature is utilized by IBM and software vendors.

Refer to section 2.5, “Connectivity” on page 48 for more information on these enhancements.

GCS Enhancements

In Release 2.2, a GCS application is allowed to exploit VM/ESA Data Spaces. This supplies significant storage constraint relief for those applications with large amounts of data and also allows for faster passing of data between CMS and GCS virtual machines.

As part of the Data Space support, GCS has extended diagnostic capabilities to handle Data Spaces. GCS now supports a GCS group consisting of XA and XC mode virtual machines, partially removing the restriction that all members of a GCS group have to be the same mode. This allows an easier migration to the new release of GCS for those applications who do not currently support VM Data Spaces and XC mode.

To enhance communications between members of a GCS group, GCS has provided a new interface to allow applications to dynamically create, delete, and retrieve a NAME/TOKEN pair. This can be used to pass information between processes running in the same virtual machine or applications running in different virtual machines. This allows applications to reduce or eliminate use of IUCV for tasks where performance is critical. It also allows recovery of a single virtual machine or task without having to bring down the entire GCS group. This can have a major positive impact on network reliability.

In addition, several serviceability enhancements have been made to GCS. For further information, refer to Chapter 6, “Group Control System (GCS) Enhancements” on page 119.

1.4 VM End of Support Dates

The number of supported releases of VM has changed dramatically over the last two years. As Withdrawal From Marketing Dates (WDFM) and End of Service Dates (EOS) were announced for the old releases of VM, new releases of VM/ESA have been coming online.

Table 1 shows the WDFM and EOS dates for current VM releases, and also some of those that recently went out of support. EMEA and Japan may have different dates.

Table 1. VM End of Support Dates

PP Number	PP Name	V.R.M.	GA Date	WDFM Date	EOS Date
5664-167	VM/SP	1.5.0	May 87	27 Dec 91	31 Dec 93
5664-167	VM/SP	1.6.0	Jan 89	16 Sep 92	30 Jun 94
5664-173	VM/HPO	1.5.0	Oct 87	27 Dec 91	31 Dec 93
5664-173	VM/HPO	1.6.0.	Jan 89	27 Dec 91	31 Dec 93
5664-308	VM/XA	1.2.1	Dec 87	27 Mar 92	30 Jun 94
5684-112	VM/ESA	1.1.0 370 Feature	Sep 90	17 Dec 93	30 Dec 94
5684-112	VM/ESA	1.1.0 ESA Feature	Mar 91	20 Dec 91	31 Dec 93
5684-112	VM/ESA	1.1.1	Dec 91	18 Dec 92	30 Dec 94
5684-112	VM/ESA	1.2.0	Dec 92	24 Sep 93	31 Mar 95
5684-112	VM/ESA	1.2.1	Jul 93	10 Jun 94	
5684-112	VM/ESA	1.1.5 370 Feature	Dec 93		
5684-112	VM/ESA	1.2.2	Jun 94		

Note: VM/ESA Release 2.0 B1 Security Feature is still marketed and supported by IBM.

1.5 Back-Level CMS Support

VM/ESA Release 2.2 continues to provide support for previous releases of CMS. This is commonly called back-level CMS support. To help migration, back-level CMS releases 7, 8, 9, and 10 are supported under VM/ESA Release 2.2. This migration support is available until the end of service of the previous production CMS release, or 24 months from the availability of VM/ESA Release 2.2 (which ever comes first). Refer to Table 1 for end of service dates for the currently supported VM releases.

Table 2 shows which levels of CMS are supported under VM/ESA. CMS Level 11 is the level for VM/ESA Release 2.2.

Table 2. VM/ESA Back-level CMS Support

VM SCP	CMS5.6	CMS6	CMS7	CMS8	CMS9	CMS10	CMS11
VM/ESA R1.0 (370)		√	√				
VM/ESA R1.0 (ESA)	√	√	√				
VM/ESA R1.1	√	√	√	√			
VM/ESA R2.0	√	√	√	√	√		
VM/ESA R2.1	√	√	√	√	√	√	
VM/ESA R2.2			√	√	√	√	√

1.6 VM/ESA Hardware Requirements

VM/ESA Release 2.2 requires the ESA/370* or the ESA/390 architecture. Processor functions, such as VM Data Spaces and Storage Key Facility (SKF), can provide improved performance, but are not mandatory.

VM/ESA Release 2.2 runs on three families of processors:

ES/9000 family	The ES/9000 family includes VM Data Spaces and SKF as standard features
ES/3090* family	The ES/3090 Models E, S, J and JH and the ES/3090-9000T are supported. The ES/3090 family includes SKF, but does not include VM Data Spaces. The IBM 3090 models available before the ES/3090 E model are not supported.
ES/4381 Models 9xE	The ESA/370 capable 4381 Models 90E, 91E and 92E include neither VM Data Spaces nor SKF.

Note:

To run VM/ESA in a LPAR or as a guest of VM/ESA on a 3090S (18S and higher), RPQ 8P1367 must be installed on the hardware.

For VM Data Spaces support in VM/ESA Release 2.2 to function, the product must be IPLed on an ES/9000 machine.

1.7 Migration Services

IBM offers migration services to customers moving from prior VM platforms to VM/ESA Release 2.2. These fee-based services can provide assistance in the migration activities. Support can be provided in any of the following areas:

- Planning, installing, and tailoring VM/ESA
- Migrating VM and VM guests
- Migrating IBM and vendor products
- Service and PTF application
- Performance tuning.

Should you require assistance or need any further details on these migration services, please contact your local IBM representative. These services may not be available in all geographies.

1.8 Planning and Installation

This section gives some hints for planning and installing VM/ESA Release 2.2. It also notes some of the enhancements made to the installation process over previous releases of VM/ESA. It is intended only to complement the planning guidelines detailed in *VM/ESA Planning and Administration*.

1.8.1 Migration Support

Many tools and options are provided with VM/ESA Release 2.2 to ease the migration steps from previous releases of VM.

ESAMIGR Support

The VM/ESA REXX/EXEC Migration Tool (ESAMIGR) is shipped along with the VM/ESA installation tapes. This EXEC is used to identify incompatibilities in REXX EXECs between VM/ESA Release 2.2 and previous releases of VM.

For detailed information on the ESAMIGR tool, refer to *VM/ESA: REXX/EXEC Migration Tool*.

Migration Tools

Several tools shipped with VM/ESA Release 2.2 may be very useful during migration time. These tools are:

- DRAWLOGO EXEC

  Creates logos for your VM/ESA operating system.

- HCPTRIO EXEC

  Converts the contents of the HCPRIO ASSEMBLE file to statements that can be included in the SYSTEM CONFIG file.

- HCPTSYS EXEC

  Converts the contents of the HCPSYS ASSEMBLE file to statements suitable for the SYSTEM CONFIG file.

- HCPRDEVS EXEC

  Examines the running system and determine what devices are attached. This can be useful where you have IPLed VM/ESA using the default SYSTEM CONFIG file and now want to code your installation-specific SYSTEM CONFIG file.

- HCPDCON EXEC

  Creates a complete SYSTEM CONFIG file based on the information from the CP control blocks of VM/ESA Release 2.2.

- HCPDSYS EXEC

  Creates a complete HCPSYS ASSEMBLE file based on the information from the CP control blocks of VM/ESA Release 2.2.

All these EXECs are shipped with filetype SAMPEXEC and can be found on user ID MAINT's 2C2 minidisk. They need to be renamed to a filetype of EXEC to make them executable. Also, the DRAWLOGO EXEC needs an XEDIT macro called X$DRWL$X XEDIT. This X$DRWL$X file is shipped on user ID MAINT's 2C2 disk with a filetype of SAMPXEDI.

For further information on these EXECs, refer to *VM/ESA Planning and Administration*.

Mixed Directory Support

Several directory options on VM releases prior to VM/ESA are no longer supported in VM/ESA. For a list of these options, refer to *VM/ESA: Conversion Guide and Notebook for VM/SP, VM/SP HPO, VM/ESA (370 Feature)* or *VM/ESA: Conversion Guide and Notebook for VM/XA SP and VM/ESA*, depending on which release of VM you are running,

However, to ease migration, it is usually more convenient to maintain a single copy of the source directory that can be used with both the new and old VM system. The mixed directory support in VM/ESA Release 2.2 allows the mixture

of directory control statements from earlier releases of VM with those for VM/ESA Release 2.2.

The MIXED option on the DIRECTXA command allows unsupported options to be specified in the VM/ESA Release 2.2 source directory. These options are ignored when the directory is put online, but listed at the terminal unless the NOMIXMSG option is also specified.

For further information on the DIRECTXA command, refer to *VM/ESA CP Command and Utility Reference*.

1.8.2 Installation

Some minor changes have been made to the install procedure of VM/ESA Release 2.2. These changes are documented here.

Distribution Options

You can have your VM/ESA Release 2.2 supplied in several ways. The method you select depends on the speed with which you need to put a system into production and also on the skill level of your systems programming staff.

System DDR: VM/ESA Release 2.2 provides an automated installation procedure that uses the INSTALL EXEC with a panel interface. The tapes are supplied in DASD Dump Restore (DDR) format. The INSTALL EXEC allows the user to select which items to load, and also where those items can be placed on DASD.

For a sample of the INSTALL panel interface, please refer to Figure 3 on page 14.

System Delivery Option (SDO): The System Delivery Option has been available since the general availability of VM/ESA. In Release 2.2, it has been enhanced to ship selected program products in a pre-built format. All program products are still shipped in the stacked-tape format used in previous releases.

Details of the products that can be ordered through SDO can be obtained from your local IBM representative. The current program product number of the VM/ESA SDO is 5654-026 which replaces 5750-ACK.

Customized Installation Service: Optional fee-based services are also available for installing VM/ESA Release 2.2. These services can provide a pre-built VM/ESA Release 2.2 system, including optional program products. Services detailed in section 1.7, “Migration Services” on page 10 can also be incorporated to help the client install, customize, and test their VM/ESA operating system.

For further information, contact your local IBM representative.

Installation Manual

With VM/ESA Release 2.2, the VM/ESA Installation Guide is no longer a manual with an IBM form number. When you order VM/ESA Release 2.2, you receive the latest copy of the Installation Guide with the VM/ESA Release 2.2 installation tapes. Any updates made to the Installation Guide are included in the copy shipped with the installation tapes.

Should you require a copy of the manual for planning purposes, it can be ordered through your IBM Support Center with PTF UM98122.

VM/ESA Release 2.2 DASD Requirements

Table 3 shows the DASD requirements to install VM/ESA Release 2.2 using the default sizes as specified by IBM. The default DASD labels are E22RES, E22W01, ...E22Wxx.

Table 3. Minimum DASD Requirements for Installation

DASD Device Type	Density	Cylinders/Blocks on one DASD	Number Required
3380	Single	0885	3
	Double	1770	2
	Triple	2655	1
3390	Single	1113	2
	Double	2226	1
	Triple	3339	1
9345	Model 1	1440	2
	Model 2	2156	2
9332 (FBA)	Model 400	360036	9
	Model 600	554800	6
9335 (FBA)	Single	804714	4
9336 (FBA)	Model 10	920115	4
	Model 20	1672881	2

Installation Changes

This section gives you an overview of some of the major installation changes in VM/ESA Release 2.2.

New MOVE2SFS EXEC MOVE2SFS is a new EXEC provided with VM/ESA Release 2.2 to assist the user in moving standard VM components from minidisk to Shared File System directories. A sample execution of the command is shown in Figure 2.

```
move2sfs
HCPWMV8456I PROCESSING COMPONENT GCS
HCPWMV8453I GCS COMPONENT COMPLETED SUCCESSFULLY
HCPWMV8456I PROCESSING COMPONENT TSAF
HCPWMV8453I TSAF COMPONENT COMPLETED SUCCESSFULLY
HCPWMV8456I PROCESSING COMPONENT AVS
HCPWMV8453I AVS COMPONENT COMPLETED SUCCESSFULLY
```

Figure 2. Sample Output from MOVE2SFS EXEC

INSTALL EXEC: The INSTALL EXEC has been enhanced to provide users with more options in loading their VM/ESA Release 2.2 system from tape.

Two new options have been added to the INSTALL panel:

- Load Shared File System (SFS)
- Load uppercase English help files.

If you do not select to load SFS, the VMSYS: VMSYSU: and VMSYSR: Shared File System server minidisks are not restored from the DDR. All the SFS function in CMS is still loaded and available, just the three default servers are not loaded and initialized. The same is true for the loading of the uppercase English help files. A sample of the new INSTALL panel is shown in Figure 3.

```
                        VM/ESA RELEASE 2.2 LOAD MENU

              ENTER 'S' TO SELECT   ('L' INDICATES ALREADY LOADED)

                     S        BASE (CP, DV, CMS, REXX, VMSES/E)
                     S        GCS
                     S        TSAF, AVS
                     S        SFS
                     S        UPPERCASE ENGLISH HELP
                     S        CP, DV SOURCE
                     S        CMS, REXX SOURCE
                     S        VMSES/E SOURCE


====>
 PF1 = HELP         PF3 = QUIT        PF4 = UNLOCK RELOAD      PF5 = NEXT
```

Figure 3. Sample INSTALL Panel

For further information on the installation procedure, refer to *VM/ESA Installation Guide* supplied with your VM/ESA installation tapes.

CMS Nucleus Changes: To conserve space in the CMS nucleus, many of the CMS QUERY and SET commands have been moved into disk-resident modules. This occurred in VM/ESA Release 2.0. The recommendation has been to put these disk-resident modules into a saved segment.

In Release 2.2 two new logical segments have been provided as part of the base code:

- CMSQRYL
  This segment contains the query and set routines that can only run with 24-bit addressing. Hence, they need to be loaded below the 16MB line.
- CMSQRYH
  This segment contains the query and set routines that can only run with 31-bit addressing. Hence, they can be loaded above or below the 16MB line.

1.9 Query levels and Diagnose Code Results

Application programmers commonly ask for the results returned by the following commands:

QUERY CMSLEVEL

```
q cmslevel
CMS Level 11, Service Level 404
```

QUERY GCSLEVEL

```
q gcslevel
VM/ESA Release 2.2, Service Level 404
```

QUERY CPLEVEL

```
q cplevel
VM/ESA Release 2.2, service level 9404
```

Diagnose X'00' Results

```
VM Type:     VM/ESA
Environment: 000000
Version code FF
Ext Layout   0000
Proc adr.    0000
User ID.     MAINT
PP bit map   7FF0000000000000
Time zone    -4
Release Inf. 020224BC
```

Chapter 2. CP Enhancements

This chapter describes the enhancements made to CP in VM/ESA Release 2.2. The topics covered are a comprehensive minidisk caching enhancement, the improved spool backup facility, scheduler enhancements, a new CP IPL option, connectivity enhancements, and many others.

2.1 Minidisk Caching Enhancement

Because of the benefits minidisk caching can provide to the end user, it has been enhanced in this release of VM/ESA to make it available to a wider audience of VM users. Some of the restrictions that applied to the usage of minidisk cache have been removed, and additional commands have been introduced to ease the management of the cache subsystem.

2.1.1 Use of Minidisk Cache Before VM/ESA Release 2.2

Prior to VM/ESA Release 2.2, minidisk data was cached only if the following criteria were met:

1. Data resided on a CMS minidisk that had been formatted in 4KB blocks.
2. Expanded storage was available on the system.
3. Data on the minidisk was read using one of the following interfaces:
   DIAGNOSE X′18′ Standard DASD I/O
   DIAGNOSE X′20′ 370 Synchronous I/O for Diagnose Support
   DIAGNOSE X′A4′ Synchronous I/O Operation for DASD in standard CMS blocksize
   DIAGNOSE X′A8′ Synchronous I/O Operation for all Devices
   DIAGNOSE X′250′ Block I/O Operations
   ***BLOCKIO** CP System Service
4. Data was not on a minidisk shared among other systems.
5. Non full-pack minidisks
6. FBA minidisks aligned on a page boundary

For detailed information on the DIAGNOSE codes specified above, refer to *VM/ESA CP Programming Services*.

Because data needed to reside in CMS formatted 4KB minidisks, the benefits of caching were not available to CMS users with disks formatted in blocksizes other than 4KB and to non-CMS users. As VSE does not use any of the above I/O interfaces, the large population of users running VSE as a guest to their VM system were excluded from the benefits of minidisk caching.

2.1.2 Enhancements in VM/ESA Release 2.2

Because of pre-Release 2.2 restrictions and requests from customers for increased function, minidisk caching has been enhanced. This section details the enhancements made to minidisk caching.

Track Caching

Minidisk caching now works on a track basis as opposed to 4KB blocks. As a result, the disks no longer need to be formatted by CMS into 4KB blocks. Assuming the data is eligible for caching (see section 2.1.3, “Minidisk Cache Hierarchy” on page 19 for further details), the data can be from a CMS user, VSE user, MVS user, or any other guest operating system supported by VM/ESA.

This allows disks that previously were not supported to benefit from caching. Also, for current users of minidisk cache, it may reduce the number of I/Os to DASD since a full track of DASD is now read, as opposed to multiple accesses of 4KB blocks.

In addition to the diagnose codes and *BLOCKIO services specified in section 2.1.1, “Use of Minidisk Cache Before VM/ESA Release 2.2” on page 17, I/O issued using the Start Sub-Channel (SSCH), Start I/O (SIO) and Start I/O Fast (SIOF) are eligible for minidisk caching also. This rounds out the support to include all current forms of VM/ESA DASD I/O. Fullpack minidisks can be cached and, though not recommended due to the possibility of data corruption, shared DASD can be cached.

Storage Enhancement

In pre-VM/ESA Release 2.2 systems, minidisk caching only occurred within expanded storage. This limitation has now been removed. Minidisk caching can now occur to either expanded storage, main storage, or a combination of both. This opens the usage of minidisk caching to all VM/ESA systems on all CPUs.

New CP Commands

New CP commands have been introduced to give the VM/ESA systems programmer and general user improved control over cached data. Two new commands have been introduced:

- SET MDCACHE
- QUERY MDCACHE

These commands allow the systems programmer to:

- Set the amount of real and expanded storage available for minidisk cache
- Query the amount of real and expanded storage in use for minidisk cache
- Set and query cache settings for a real device or minidisk
- Purge the cache data for a particular device or minidisk
- Set and query the overall state of the system level minidisk cache.

Supported DASD for Minidisk Cache

The following devices are supported for Minidisk Cache:

- Count Key Data (CKD and ECKD) devices:
  - 3380
  - 3390
  - 9345
- Fixed Block Architecture (FBA) devices:
  - 3370
  - 9332
  - 9335
  - 9336

  Note: FBA DASD must still be page aligned to be eligible for minidisk caching. This means the number of blocks defined must be divisible by eight

(multiples of 4096 bytes = 1 page) and the first block starts on a 4K boundary. This is the only current restriction for minidisk caching.

2.1.3 Minidisk Cache Hierarchy

Before discussing the items specified above in more detail, we need to discuss the concept of minidisk cache hierarchy.

A caching hierarchy has been introduced to determine whether data read from DASD should be cached or not. The hierarchy is made up of three levels:

- System level
- Real DASD device level
- Minidisk level.

For the data to be eligible for caching, all three levels must be enabled. Unlike previous releases of VM/ESA, the users now have control over whether their minidisks should be cached. Users also can query information regarding their data that is cached.

System Level Caching

The system level is the highest level of control for minidisk cache. This controls whether minidisk caching is operational on the current installation. The system level is set using the CP SET MDCACHE SYSTEM command (see section 2.1.4, “SET MDCACHE Command” on page 20). The current setting of the system level cache can be queried using the CP QUERY MDCACHE SYSTEM command (see section 2.1.5, “QUERY MDCACHE Command” on page 22). There are further controls to set minidisk caching for main storage (STORAGE) and expanded storage (XSTORE).

The user requires a class B privilege virtual machine to be able to query or set the minidisk cache at a system level. The default setting is on.

Note: Issuing both SET MDCACHE XSTOR OFF and SET MDCACHE STOR OFF or issuing SET MDC SYSTEM OFF prevents caching from occurring on the system. However, issuing SET MDC SYSTEM OFF immediately purges all cache data, disables caching on the system, cleans up all internal minidisk cache structures, and releases all storage used by caching. Issuing SET MDC XSTORE OFF and SET MDC STOR OFF causes a gradual reduction in cache data and does not clean up internal structures or storage. Furthermore, caching is still considered enabled for the system.

Real Device Level Caching

Real device level caching is the second level of control for minidisk caching. If minidisk caching is set on at a system level, then the cache eligibility of a particular device can be changed using the CP SET MDCACHE *rdev* command. This command has four options as explained here:

DFLTON Default On enables caching for a real device. However, it still allows for caching to be turned off at a minidisk level. All minidisks on this device have minidisk caching set to ON, unless the user explicitly turns minidisk caching off for a specific minidisk or range of minidisks. This is the default.

DFLTOFF Default Off disables caching for a real device. However, it still allows for caching to be turned on at a minidisk level. All minidisks on this device have minidisk caching set to OFF, unless the user explicitly turns minidisk caching on for a specific minidisk or range of minidisk.

OFF OFF disables minidisk caching for this real device. It cannot be overridden by the user at a minidisk level. No minidisks are cached from this device.

FLUSH FLUSH clears, from minidisk cache storage, all data from this device.

Minidisk Level Caching

Minidisk level is the third and final layer of control for minidisk caching. At this level, caching can be turned on or off for a particular minidisk by the system programmer or the user who owns that disk. If caching is ON for the system, and the caching option on the real device is not set to OFF, then the cache eligibility of the minidisk can be changed using the CP SET MDCACHE MDISK command. If caching is disabled at a higher level, then the command will fail.

This command has three options which are explained here:

ON ON enables caching for a minidisk. All data on this minidisk is cached.

OFF OFF disables caching for a minidisk. Data on this minidisk is not cached.

FLUSH FLUSH clears, from minidisk cache storage, all data from this minidisk.

If no option is specified and neither MDC nor NOMDC were selected on the MINIOPT directory control statement, the default is taken. The default option means that caching is ON for a minidisk whose real device has caching set to DFLTON. Caching is OFF for a minidisk whose real device has caching set to DFLTOFF or OFF.

2.1.4 SET MDCACHE Command

The CP SET MDCACHE command is available to both class B and class G privilege users. The options available to each user depend on this class. Figure 4 shows the command syntax for the class B privilege user.

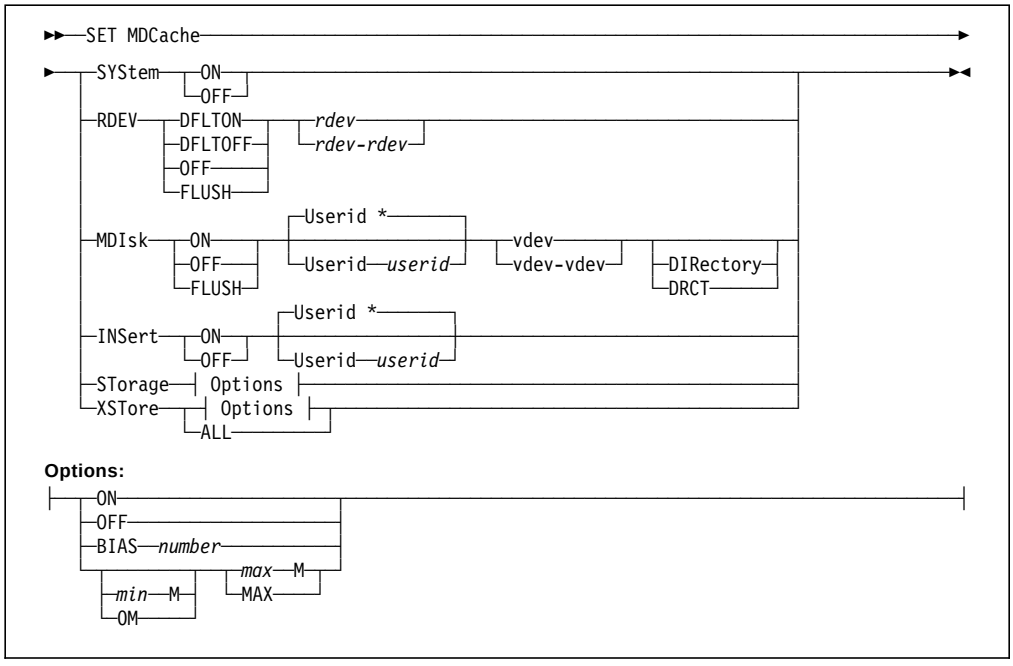


Figure 4. CP SET MDCACHE Command Syntax (Class B Privilege)

Users with privilege class B can:

- Change minidisk cache settings for the entire system, for a real device, or for an active minidisk.
- Flush the cache of data from a real device or an active minidisk.
- Change a user's ability to insert data into cache.
- Control usage of main and expanded storage, and bias the minidisk cache arbiter toward minimum and maximum storage values with the BIAS parameter.

Figure 5 shows the command syntax for the class G privilege user.

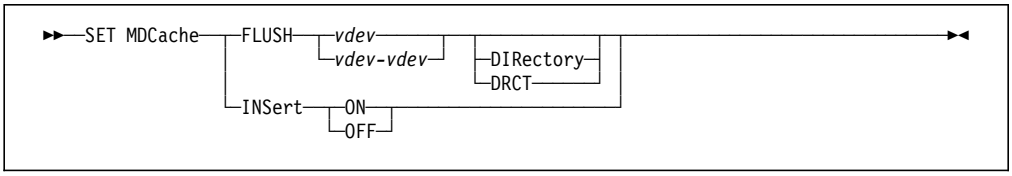


Figure 5. CP SET MDCACHE Command Syntax (Class G Privilege)

Users with privilege class G can:

- Purge data in minidisk cache on an active minidisk defined in their directory.
- Change the ability to insert data into the cache.

Notes:

1. If the amount specified with the SET MDCACHE STORAGE is greater than the amount available, the amount specified is rounded down to the amount available.
2. If SET MDCACHE STORAGE *max*M is issued when *max* is greater than the current size of the minidisk cache area in expanded or main storage, a gradual decrease in the use of expanded storage or main storage for CP paging occurs until the minidisk cache attains the new larger size.

   This does not cause a rapid migration of paging data to auxiliary storage. Similarly, when *max*M is less than the current size of the minidisk cache, a gradual reduction in the use of minidisk cache occurs until the minidisk cache attains the new size.
3. If you run operating systems as guests to VM/ESA Release 2.2 that are capable of caching data, and the cached disks are available to both first and second level systems, **minidisk caching should only take place on one system**. If caching occurs on both systems, data may not be reflected correctly between the two systems.
4. If the directory option is not used, the user must be logged on.
5. If your installation uses expanded storage purely for minidisk caching, you may consider returning this storage to main storage (if your processor allows), as minidisks can now be cached in main storage.

For more detailed information on the SET MDCACHE command, refer to *VM/ESA CMS Command Reference*.

2.1.5 QUERY MDCACHE Command

Like the CP SET MDCACHE command, QUERY MDCACHE has different options, depending on whether the user has class B privilege or class G privilege. Figure 6 shows the command syntax for the class B privilege user.

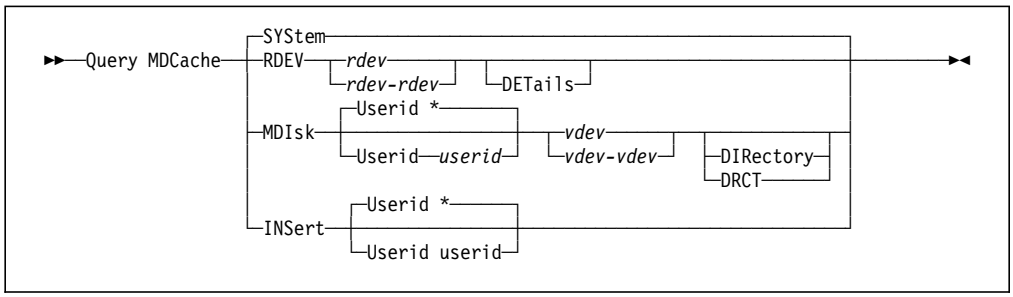


Figure 6. CP QUERY MDCACHE Command Syntax (Class B Privilege)

Privilege class B users can:

- Query minidisk cache settings for the entire system, for a real device, or for an active minidisk defined in the directory.
- Query a user's ability to insert data into the cache.

Figure 7 shows the command syntax for the class G privilege user.

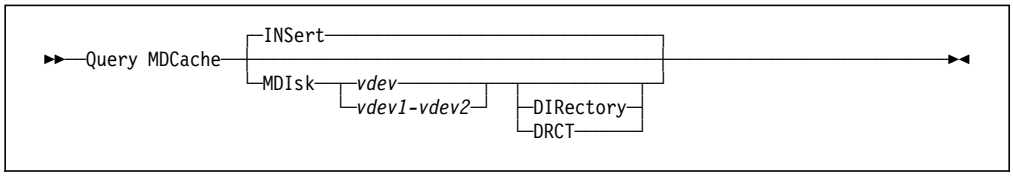


Figure 7. CP QUERY MDCACHE Command Syntax (Class G Privilege)

Users with class G privilege can query their ability to insert data into the cache.

For more information on these commands, refer to *VM/ESA CP Command and Utility Reference*.

2.1.6 Configuration Changes for Enhanced Minidisk Caching

The enhancements made to minidisk caching have caused changes to CP directory statements and also to the RDEVICE macro. These enhancements are described in the following sections.

Directory Statements

The enhanced minidisk caching has changed and added options for the CP directory. They are discussed in the following sections.

Directory MINIOPT Statement: The MDC and NOMDC options to the MINIOPT directory control statement have changed, reflecting the support by the VM/ESA Release 2.2 enhancement to minidisk caching. The new syntax of the MINIOPT statement is shown in Figure 8 on page 23.

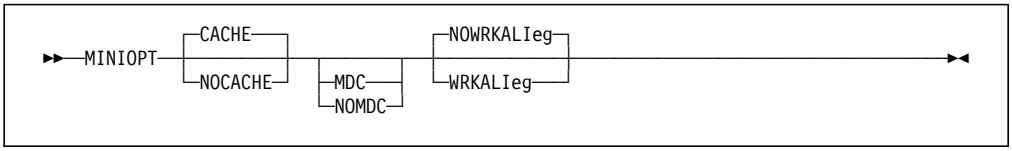


Figure 8. MINIOPT Directory Control Statement

The WRKALIeg/NOWRKALIeg options are detailed in section 2.8, “Working Allegiance (MVS Sysplex DASD Sharing)” on page 64.

The MDC options provide the following functions:

MDC Specifies that the minidisk is cached when caching is set to DFLTON or DFLTOFF for the real device.

NOMDC Specifies that the minidisk is not cached.

If neither MDC nor NOMDC is specified, the minidisk will use the minidisk cache when caching is set to DFLTON for the real device. For further information on the MINIOPT directory control statement, refer to *VM/ESA Planning and Administration*.

Directory Option Statement: Minidisk cache uses an algorithm called the cache fair share algorithm. This algorithm attempts to give all users of the system equal and fair access to the minidisk cache subsystem. However, there are times you may want certain user IDs to have better access to the minidisk cache (for example, you may want a second-level production VSE system to have better access to the minidisk cache than an OfficeVision/VM user).

As a result, the NOMDCFS option (NO MiniDisk Cache Fair Share) can be specified in the OPTION directory control statement of the user ID. This option specifies that the virtual machine can use minidisk cache at a rate that is not limited by the fair share limit. This allows virtual machines with a high I/O rate to get the full benefit of minidisk cache.

This option is not new to VM/ESA Release 2.2, but can be of increased benefit as now guest operating systems minidisks can be cached.

RDEVICE Option Change

The RDEVICE macro of the VM/ESA Release 2.2 SYSTEM CONFIG file has been updated to reflect the changes made to support full track minidisk caching. The new syntax of the RDEVICE macro is illustrated in Figure 9.

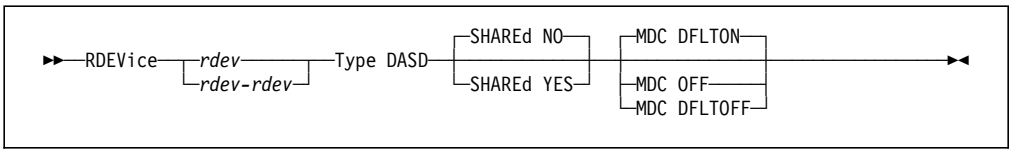


Figure 9. RDEVICE Macro for Minidisk Cache Support

The MDC options were explained in section 2.1.3, “Minidisk Cache Hierarchy” on page 19.

For more information on the RDEVICE macro, refer to *VM/ESA Planning and Administration* .

A similar change has been made to the CP SET RDEVICE command to add the options for the minidisk cache enhancement. For further information, refer to *VM/ESA CP Command and Utility Reference*.

INDICATE LOAD

The class E and class G privilege commands have been changed to reflect the fact that minidisk caching can now occur to both real and expanded storage. The MDC section of the INDICATE LOAD command now displays the total READ and WRITE count and hit ratio for all the minidisk cache storage.

Please refer to Figure 11 on page 25 for a sample of the CP class E privilege command. For further details on this command, refer to *VM/ESA CP Command and Utility Reference*.

Changed MONITOR Records

To support the new enhancements in VM/ESA Release 2.2, several monitor records were modified, and new records have been added. These updates are shown in Table 11 on page 157.

2.1.7 Minidisk Cache Sample Command Output

This section shows sample commands issued by a class G privilege user, ORGANK, (Figure 10) and a class B privilege user, MAINT, (Figure 11).

```
ind load

AVGPROC-000% 01 AVGVEC-000% 01
XSTORE-000000/SEC MIGRATE-0000/SEC
MDC READS-000001/SEC WRITES-000001/SEC HIT RATIO-100%
STORAGE-000% PAGING-0000/SEC STEAL-000%
Q0-00000 Q1-00001           Q2-00001 EXPAN-001 Q3-00000 EXPAN-001
Ready; T=0.01/0.01 12:12:14

q mdcache

Minidisk cache inserts ON for ORGANK
Ready; T=0.01/0.01 12:12:26

q mdcache insert

Minidisk cache inserts ON for ORGANK
Ready; T=0.01/0.01 12:13:16
```

Figure 10. Sample Minidisk Caching Commands (Class G Privilege)

```
ind load

AVGPROC-000% 01 AVGVEC-000% 01 VECTOR-00000
XSTORE-000000/SEC MIGRATE-0000/SEC
MDC READS-000001/SEC WRITES-000001/SEC HIT RATIO-086%
STORAGE-000% PAGING-0001/SEC STEAL-000%
Q0-00000(00000)                          DORMANT-00035
Q1-00000(00000)           E1-00000(00000)
Q2-00000(00000) EXPAN-001 E2-00000(00000)
Q3-00001(00000) EXPAN-001 E3-00000(00000)
PROC 0000-000% VEC-000%
LIMITED-00000
Ready; T=0.01/0.01 12:16:46

q mdcache system

Minidisk cache ON for system
Storage MDC min=0M max=999M, usage=0%, bias=1.00
Xstore MDC min=0M max=128M, usage=13%, bias=1.00
Ready; T=0.01/0.01 12:17:10

set mdcache system off

Minidisk cache OFF for system
Ready; T=0.01/0.01 12:17:35

q mdcache system

Minidisk cache OFF for system
Ready; T=0.01/0.01 12:17:39

set mdcache system on

Minidisk cache ON for system
Storage MDC min=0M max=999M, usage=0%, bias=1.00
Xstore MDC min=0M max=128M, usage=0%, bias=1.00
Ready; T=0.01/0.01 12:17:46

q mdcache rdev dad

Minidisk cache DFLTON for 0DAD
Ready; T=0.01/0.01 12:18:14

set mdcache rdev off dad

Minidisk cache OFF for 0DAD
Ready; T=0.01/0.01 12:18:56
```

Figure 11. Sample Minidisk Caching Commands (Class B Privilege)

2.1.8 Migration Considerations

With this enhancement, data on most disks used by CMS and guest operating systems is now eligible for minidisk caching. You should evaluate which disks within your operation would benefit most from minidisk caching.

There may be some minidisks that are poor candidates for minidisk caching. In previous releases of VM/ESA, these minidisks may not have been eligible for minidisk caching due to the type of I/O or their format. With several restrictions now having been lifted, it would be worthwhile to turn minidisk caching off for

these devices; for example, VSE paging packs. These packs may, by default, have minidisk caching turned on.

Some minidisks may contain numerous small files. Previously, these minidisks had to be formatted in 4KB blocks to be eligible for caching. With this restriction now removed, it may be a benefit to reformat these disks to a smaller blocksize.

You should also evaluate your usage of expanded storage. If your system runs on a processor capable of defining expanded storage as main, this may allow you to free unused expanded storage for main storage requirements.

2.1.9 Guest Performance Improvements

Minidisk caching uses main or expanded storage to reduce DASD read I/Os. Accordingly, the degree of performance benefit depends on the extent of DASD read activity in the workload and the amount of storage that is made available. Laboratory measurements using VSE guest workloads have shown elapsed time reductions of up to 50% and processor capacity improvements of up to 7% with minidisk caching.

Details of this and other performance improvements can be found in *VM/ESA Release 2.2 Performance Report*, available through your normal IBM channels.

2.2 SPOOL Backup Enhancements - SPXTAPE

SPXTAPE is designed to make SPOOL backup faster, less error prone, easier to manage, and less disruptive to your system. This section contains an overview of SPXTAPE and details on how you can use it.

To give you a better understanding of the various parameters used with SPXTAPE, you should be familiar with what kind of files are stored in VM/ESA′s spool. First of all, there are the *standard* spool files, they are the Reader, Printer, Punch, and Console files, and are typically seen in a user′s RDRLIST. The second kind are called the *System Definition Files* or SDF′s. In these files, data is stored which enhances the system. NSS (saved systems), NLS (national languages), IMG (printer support), and TRF (system trace files) are SDF files. DMP is the final kind of spool file, and is not supported by SPXTAPE.

SPXTAPE is a new CP command introduced in VM/ESA Release 2.2. It provides up to ten times improvement in performance over the SPTAPE command. SPXTAPE has been designed to cope with very large amounts of spool data and to provide good performance. Often, backup times are reduced enough to consider a daily spool backup.

The following is a summary of the highlights of SPXTAPE:

- Significant reduction in elapsed time to dump or load files
- Reduction in number of tapes required
- Log files used to track files dumped, rather than screen output
- Improved error handling
- Toleration of standard tape labels
- Tape drives are attached to the user′s virtual machine
- Multiple tape drives may be used for single operation

- Dynamic addition and/or reduction in tape drives during operation
- Flexible file selection ability
- NODUP option prevents loading of duplicate files
- G class users may use SPXTAPE to process their own spool files
- Allows files to span more than one tape.

2.2.1 SPXTAPE Compared to SPTAPE

Until VM/ESA Release 2.2, SPTAPE was the only method to specifically backup and restore system spool files. The performance of SPTAPE, although adequate for the 9900 total spool limit in the VM/SP environment of ten years ago, is unable to provide for the very large spool environment typical of today′s VM/ESA systems. In many VM/ESA systems today, it is not uncommon to have more than 50,000 spool files. Using SPTAPE, it may take over 24 hours to dump the entire spool to tape and restore it. In today′s environment of providing high availability and continuous service, having a major system unavailable for a full day or more is unacceptable.

Because of the limitations of SPTAPE′s performance, many installations have been poorly equipped to deal with spool file management. Issues, such as migrating spool data from one set of DASD volumes to another, have been a major obstacle to manageability. It has also been impractical to have any form of spool backup, aside from NSS and IMG files, available in case of loss of data.

SPXTAPE operates as a general user or system operator command. Its function is implemented completely in CP. It can be used by class G users to process spool files that are owned by that user ID, including trace files. Class E users can process SDF′s, and class D users can process all spool files. SPXTAPE requires that real tape drives be attached to a user′s virtual machine before starting a SPXTAPE operation. This gives class G users control over the device. With SPTAPE, it was possible to leave a tape mounted. Operators could mistakenly assign the drive to another user ID and a valuable spool file dump could be overwritten.

SPXTAPE is not media compatible with SPTAPE; that is, tapes produced with SPXTAPE cannot be processed by SPTAPE, and tapes produced with SPTAPE cannot be processed by SPXTAPE. Therefore, if you need to transfer dumped spool files between VM/ESA Release 2.2 and earlier releases of VM/ESA or other versions of VM, you must use SPTAPE. The SPTAPE command is still available on VM/ESA Release 2.2.

Almost all of the CPU used by SPXTAPE is accounted for and scheduled on the issuing user ID. This is in contrast to the implementation of SPTAPE, where most of the resource usage was attributed to SYSTEM.

SPXTAPE is the preferred method for backup of spool files and system data files. There are some slight differences in function of similar SPTAPE and SPXTAPE commands:

- Tape position default is RUN in SPXTAPE; in SPTAPE it is LEAVE.
- Using SPXTAPE, a tape drive must be attached to the user′s virtual machine.
- SPXTAPE expects a virtual device number rather than the real address. Q V TAPES command is used to check devices.

- SPXTAPE identifies which files have been processed in log files, rather than with a response to the console.
- SPXTAPE provides additional selection criteria to identify the files that are to be dumped or loaded.
- Like SPTAPE, SPXTAPE does not dump open files; for example, a reader being peeked at by a user or the CP dump file. Unlike SPTAPE, open files are reported, in this case through the Volume Log.
- A range of devices may be specified when SPXTAPE is invoked. If a device does not exist in a range, SPXTAPE selects devices that exist within the range specified.
- SPXTAPE SCAN returns meaningful information within each Volume Log. See Figure 22 on page 38.
- SPXTAPE END allows selectable tape disposition.

2.2.2 SPXTAPE Command Syntax

There are five different SPXTAPE operations:

- SPXTAPE DUMP
- SPXTAPE LOAD
- SPXTAPE SCAN
- SPXTAPE END
- SPXTAPE CANCEL

The command syntax for SPXTAPE is shown in the following figures: Figure 12 shows SPXTAPE END and SPXTAPE CANCEL, Figure 13 shows SPXTAPE DUMP, and Figure 14 shows SPXTAPE LOAD and SPXTAPE SCAN.

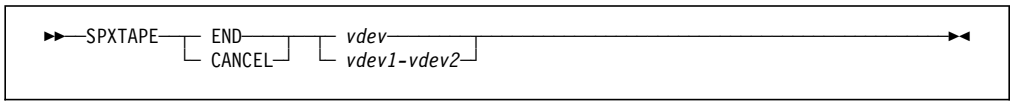
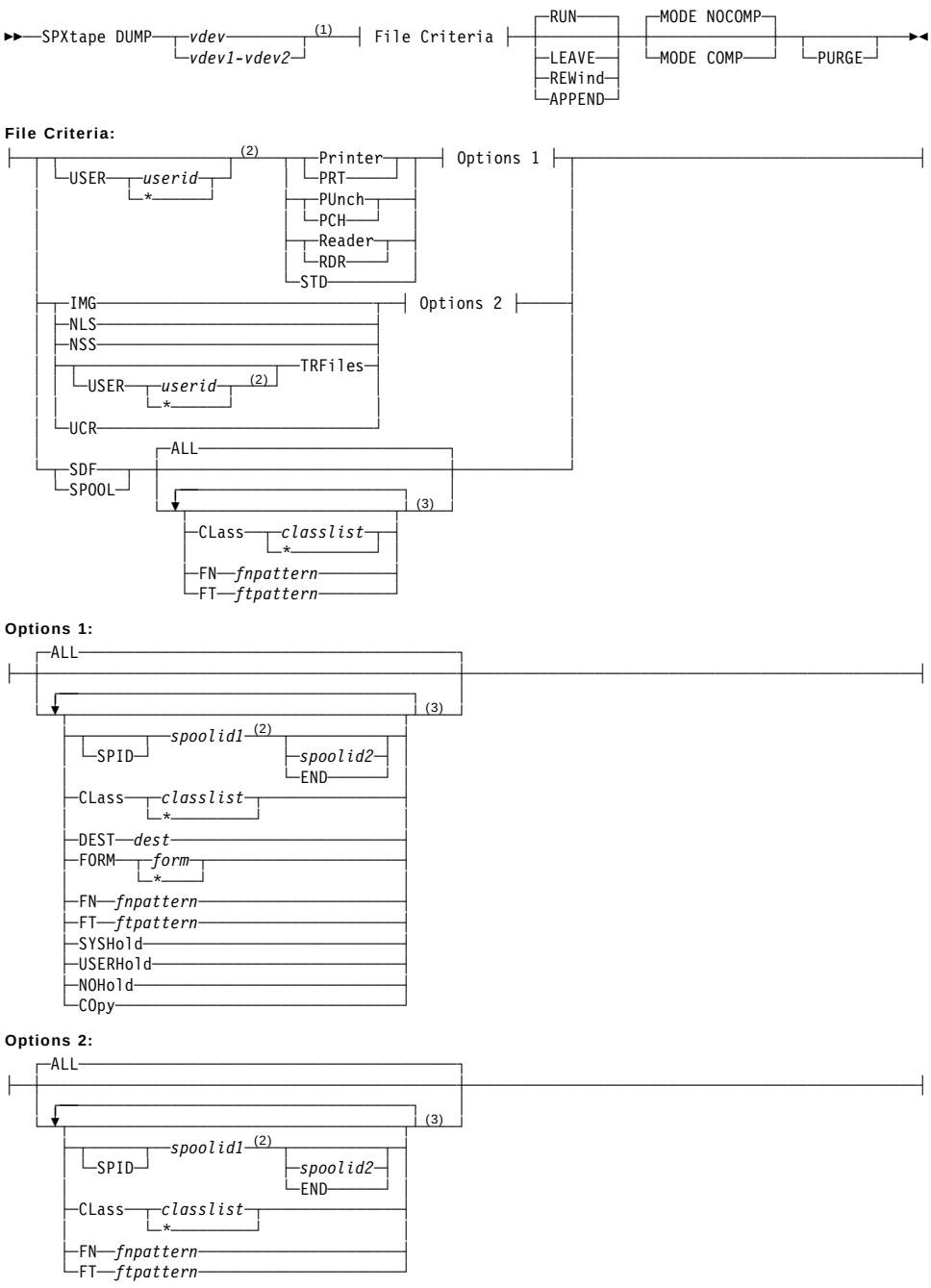


Figure 12. CP SPXTAPE END and SPXTAPE CANCEL Command Syntax

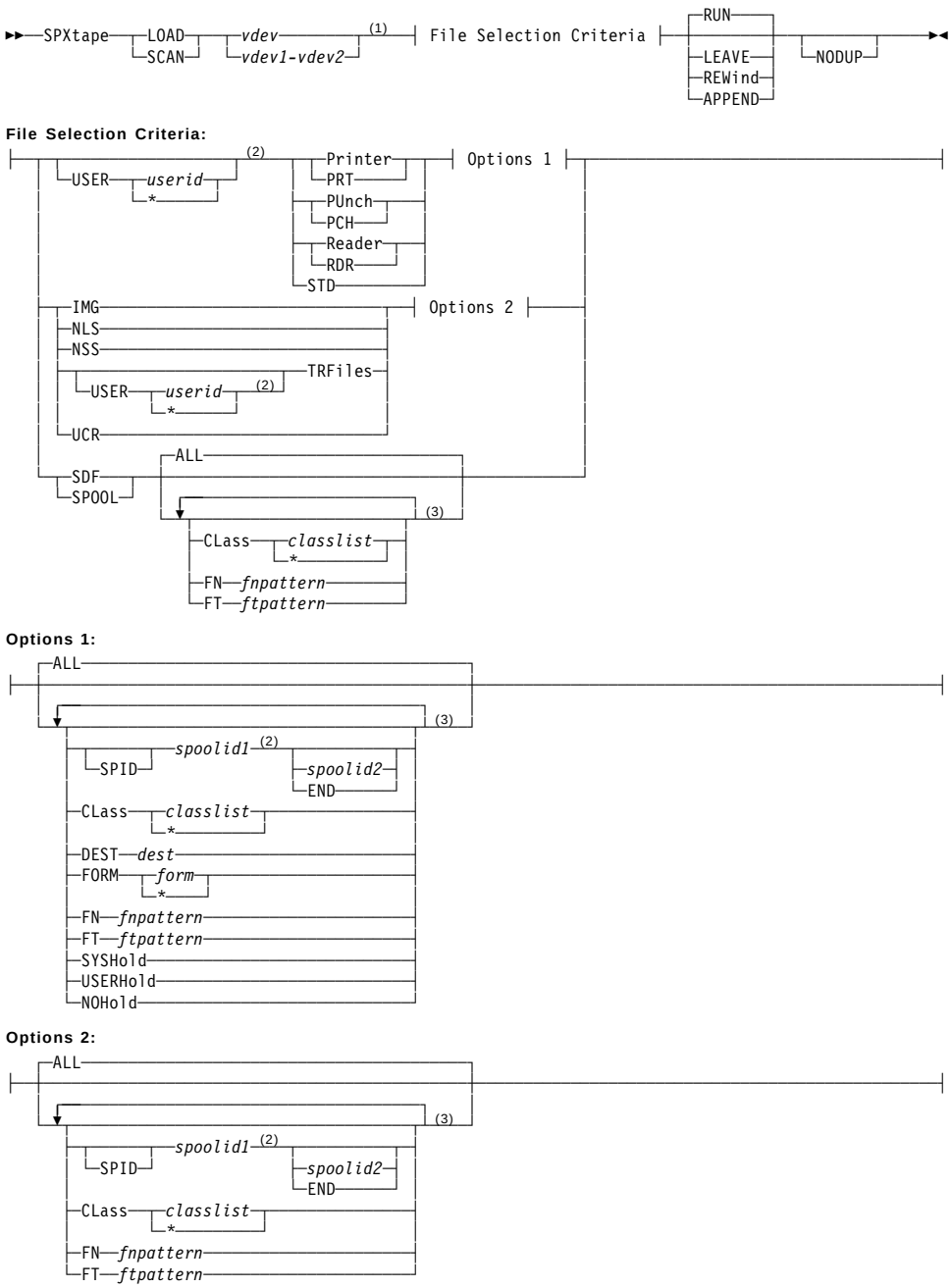
The file selection criteria for SPXTAPE LOAD and DUMP are the same except that COPY refers to Cross System Extension (CSE) copy spool files during dumping and does not apply to the LOAD command.



Notes:

1. You may use abbreviated syntax for adding devices to the SPXTAPE DUMP command. See section "Dynamic Modification of Tape Drives" on page 33.
2. For a class D or E user, USER *userid* is required if you specify *spoolid* with PRINTER, PRT, PUNCH, PCH, READER, RDR, STD, or TRF files.
3. You can specify selection operands in any order, but you may specify each only once.

Figure 13. CP SPXTAPE DUMP Command Syntax



Notes:

1. An abbreviated syntax is permitted for adding devices to a previous SPXTAPE DUMP command. See section "Dynamic Modification of Tape Drives" on page 33 for further details.
2. For a class D or E user, USER *userid* is required if you specify *spoolid* with PRINTER, PRT, PUNCH, PCH, READER, RDR, STD, or TRF files.
3. You can specify the file attribute selection operands in any order, but you may specify each operand only once.

Figure 14. CP SPXTAPE LOAD and SPXTAPE SCAN Command Syntax

2.2.3 File Selection

SPXTAPE allows you to discriminate which files are selected for processing allowing you to vary the scope of your backup. This section discusses the details of file selection.

Flexible File Selection

SPXTAPE allows for the files to be selected on the following basis:

- User ID
- SDF, STD, and types of each
- Spool ID
- Class
- Destination
- Form type
- Filename
- Filetype
- SYSHOLD, USERHOLD, NOHOLD setting
- CSE complex COPY or original.

Some examples of SPXTAPE File selection are shown in Figure 15.

```
SPXTAPE LOAD 181 RDR CLASS AB
SPXTAPE LOAD 181 USER TIMOTHY FN %%%%S FT *EXEC* LEAVE
SPXTAPE LOAD 181 NSS ALL RUN
SPXTAPE DUMP 181 PRT FORM STANDARD RUN
SPXTAPE DUMP 181 STD SYSHOLD REW
```

Figure 15. CP SPXTAPE Commands to Select Files to be Processed

In these examples:

- The first command issued starts a SPXTAPE LOAD of all class A and class B files.
- The second command uses SPXTAPE LOAD to load every file for user TIMOTHY with a five-character filename ending in *s*, and a filetype of four to eight characters containing the word *EXEC*. The LEAVE option positions the tape at the end of the last file processed.
- The third command uses SPXTAPE LOAD to load all NSS files back to the spool and unloads the tape from the drive.
- The fourth command uses SPXTAPE DUMP to dump all PRT files that are form STANDARD to tape and unloads the tape from the drive.
- The last SPXTAPE command DUMPS all RDR, PRT, or PUN spool files that are in SYSHOLD status, then rewinds the tape leaving it mounted.

SPXTAPE Processing of Duplicate Files

One of the SPTAPE problems when loading a spool system, is the potential creation of duplicate spool files when a spool restore is performed. This could happen when the restore is restarted due to system failure or other interruptions. You could also have a problem when you wish to migrate NSS files by tape between system images, having different levels of CMS and not wanting to back-level newer versions. Or, if a particular user requires their entire set of reader files to be restored (they forgot which file they discarded), every file for that user would need to be restored before the unique file is located.

SPXTAPE adds the NODUP option to the SCAN and LOAD options. NODUP bypasses loading files from tape which are a duplicate of those already on the system. The term duplicate means different things depending on the type of file.

For standard spool files and trace files, duplicate means that the name, type, dist, and all other attributes, except spoolid, are exactly the same. The time stamp must match to the microsecond. Regarding system data files other than trace files, duplicate means that the filename, filetype, and class of the tape file are identical to one on the system spool. In addition, for NSS files with class R (restricted) or class A (active) only the same filename and filetype are needed to be considered a duplicate. You simply cannot have two GCS′s with the same filename and filetype, one class A and the other class R.

Duplicate files are noted in the Volume Log when the NODUP option is used. Moving files to different user IDs, or changing their attributes before the load, such as its spool class, removes the duplicate status.

It is important to understand that using the NODUP option causes the entire spool queue to be examined. If 1000 files are being loaded to a system with 10,000 files, 10 million compares are required. This uses significant resource.

The NODUP option is best suited to system data file and single user restores.

2.2.4 Tape Processing

This section discusses SPXTAPE′s use of Improved Data Recording Capability (IDRC), standard tape labels, dynamic tape drive modification, the Append option of SPXTAPE, error recovery, and multi-volume files.

IDRC and SPXTAPE

By default, SPXTAPE does not use the IDRC feature of IDRC-enabled 3480 and 3490 devices. IDRC must be specifically invoked on the SPXTAPE DUMP command by using the MODE COMP option which is similar to the DDR MODE COMP option. Using the MODE COMP option, software compression is selected if IDRC (hardware compaction) is not available.

Even without using IDRC hardware compaction, SPXTAPE′s software compaction significantly increases a tape′s capacity, reducing the number of tapes required to complete a spool backup.

Tape Labels

SPXTAPE tolerates tapes with standard tape labels. Before the SPXTAPE command attempts to dump to or load from a tape, it reads the first record on the tape to determine if it is a standard tape label. If the first record is a standard tape label, it is skipped over and processing of SPXTAPE begins after the label. Neither EOF1 nor EOV1 trailers are written.

Dynamic Modification of Tape Drives

While the SPXTAPE command is active, you may modify the number of tape drives by either adding or removing devices. An example of modifying a SPXTAPE DUMP command is shown in Figure 16.

```
SPXTAPE DUMP 181-182 STD
SPXTAPE DUMP 183-188
SPXTAPE END 182
SPXTAPE CANCEL 183
```

Figure 16. CP SPXTAPE DUMP Command - Example of Modification of Devices

In this example:

- The first command issued starts a SPXTAPE DUMP of all standard spool files (reader, printer, punch). Assuming CP privilege class D authority, the scope of the command would be system wide.
- The second command adds devices 183 to 188 to the tape drives to be used.
- The third command removes device 182 from the active dump as soon as the actively dumping spool file has finished.
- The last command removes 183 from the dumping process immediately, without waiting for the file to finish dumping. All information of files dumped to this drive are removed from the output. If an active tape is available, files dumped on a canceled drive are dumped again on an available drive.

Multiple Volume Files

SPXTAPE supports very large spool files. It processes files that are too large to fit on a single tape cartridge or reel. A file that exceeds the capacity of one physical tape can span as many physical tapes as is required to hold it.

To load a multi-volume file using the SPXTAPE LOAD command, only a single logical SPXTAPE LOAD command should be used to restore the file. The tapes may be mounted in any order, on multiple tape drives if you choose, and SPXTAPE rebuilds the file correctly.

Error Recovery

Error recovery in SPXTAPE has changed from SPTAPE. SPXTAPE is designed to cope with errors and to continue loading, dumping, or scanning, so that as much data as possible may be processed or recovered.

All selected files are dumped if the control information is readable. Even if some of a file′s data is not readable, the readable data is still processed. If tape media or tape drives fail, SPXTAPE attempts to recover. If it cannot, then it stops writing to or reading from the tape. The drive is still active and will start processing again if a new tape is mounted on it. A SPXTAPE CANCEL or END

command can be used to take a drive out of this SPXTAPE operation. The operator can add new drives to the process at any time with no loss of data, if at least one drive remains actively processing data.

If an error occurs and a file cannot be dumped or loaded, the error is noted in the log file, the file is skipped, and processing continues with the next file.

If a tape error occurs during SPXTAPE DUMP, the tape is disregarded by SPXTAPE. A message, HCP1844E, is generated which asks the SPXTAPE operator to do one of the following:

- Discard the tape and mount new tape media to continue with the DUMP.
- Leave the drive not ready if other devices are active.
- Dynamically add more devices.

If a tape error occurs during SPXTAPE LOAD, a successful read is attempted up to a maximum of ten times. If successful, the LOAD process continues. If unsuccessful, the log will show that some files have been skipped. The exact number of skipped files cannot be determined, since an error could encompass many files. Processing of other tape volumes is unaffected by errors on any single tape.

2.2.5 Using SPXTAPE

You have already seen the SPXTAPE command in various formats in previous sections. In this section, we will show you some details of SPXTAPE operation.

SPXTAPE DUMP and the Append Option

The APPEND option of SPXTAPE DUMP allows for multiple commands to be issued pertaining to a single SPXTAPE operation. This may be referred to as a *logical* SPXTAPE command. This allows you to select files individually or in groups to be dumped, loaded, or scanned on the same set of tapes. SPXTAPE processes asynchronously, so there is no need to wait for the command to finish before issuing a further SPXTAPE command; this reduces the complexity of any EXEC procedure that your installation may write to use SPXTAPE.

Using the APPEND option prevents the SPXTAPE DUMP process from completing when it has processed the files that have been selected. Instead, SPXTAPE waits for another DUMP command that uses the same range of tape drive addresses that either selects additional files or does not use the APPEND option. This option is useful when you cannot, with a single command, specify all the files you intend to dump.

An example of some SPXTAPE DUMP commands using the APPEND option is shown in Figure 17 on page 35. When using the append option, the number of files is a running total and may not reflect every file dumped until the process has ended. Also, the percentage of completion indication may jump to a lower percentage as the number of pages added to the dump increases through each addition to the logical SPXTAPE command.

```
spool cons start to *
Ready; T=0.01/0.01 16:32:09
q sdf
FILES: 0014 IMG, 0040 NSS,   NO UCR, 0003 TRF,   NO NLS
Ready; T=0.01/0.01 16:32:13
q v 181
TAPE 0181 ON DEV  0B40 3490 R/W SUBCHANNEL = 003E
Ready; T=0.01/0.01 16:32:31
spxtape dump 181 img append
SPXTAPE DUMP   INITIATED ON VDEV 0181
Ready; T=0.01/0.01 16:32:53
SPXTAPE DUMP WAITING    ON VDEV 0181
TAPE NUMBER:         0181-001
FILES PROCESSED:            9
SPOOL PAGES:              522
spxtape dump 181 nss append
SPXTAPE DUMP   INITIATED ON VDEV 0181
Ready; T=0.01/0.01 16:33:12
DUMPING  0181     :        14  FILES,  PAGES        537    4% COMPLETE
DUMPING  0181     :        18  FILES,  PAGES      2,154   19% COMPLETE
DUMPING  0181     :        22  FILES,  PAGES      3,794   33% COMPLETE
DUMPING  0181     :        30  FILES,  PAGES      5,506   49% COMPLETE
DUMPING  0181     :        37  FILES,  PAGES      6,994   62% COMPLETE
DUMPING  0181     :        44  FILES,  PAGES      8,339   74% COMPLETE
DUMPING  0181     :        50  FILES,  PAGES      9,907   88% COMPLETE
SPXTAPE DUMP WAITING    ON VDEV 0181
TAPE NUMBER:         0181-001
FILES PROCESSED:            52
SPOOL PAGES:             11,157
spxtape dump 181 trf rew
SPXTAPE DUMP   INITIATED ON VDEV 0181
Ready; T=0.01/0.01 16:35:14
DUMPING  0181     :        57  FILES,  PAGES     11,170  100% COMPLETE
RDR FILE 0049 SENT FROM VMAINT   CON WAS 0049 RECS 0081 CPY  001 A NOHOLD NOKEEP
SPXTAPE DUMP COMMAND COMPLETED  ON VDEV 0181
TIME STARTED:        16:32:53
TIME ENDED:          16:35:14
TAPE COUNT:               001
```

Figure 17. CP SPXTAPE DUMP with APPEND Examples

QUERY VIRTUAL TAPES Command

Since tape drives must be attached to the user ID performing the SPXTAPE operation, the Q V TAPES command was enhanced to better show the status of the real device. The output of the Q V TAPES command includes information on the operation (DUMP, LOAD, SCAN) or the status (ACTIVE, APPEND, or MOUNT). Figure 18 on page 36 shows examples of a DUMP operation.

```
spxtape dump 181 nss all append
SPXTAPE DUMP   INITIATED ON VDEV 0181
Ready; T=0.01/0.01 14:22:55

q v tape
TAPE 0181 ON DEV  0B4A 3490 R/W SUBCHANNEL = 003E SPXTAPE DUMP APPEND
Ready; T=0.01/0.01 14:22:59

DUMPING  0181     :         4  FILES,  PAGES      1,617   15% COMPLETE
 ...
spxtape dump 181 img rew
Ready; T=0.01/0.01 14:23:21

q v tape
TAPE 0181 ON DEV  0B4A 3490 R/W SUBCHANNEL = 003E SPXTAPE DUMP ACTIVE
Ready; T=0.01/0.01 14:23:24

DUMPING  0181     :         9  FILES,  PAGES      3,233   28% COMPLETE
 ...
```

Figure 18. CP QUERY VIRTUAL TAPES Command Output

Using SCAN and LOAD

Some examples of CP SPXTAPE SCAN commands are shown in Figure 19 and Figure 20. Please note that in Figure 19, the output to the screen from SPXTAPE SCAN is similar to the SPXTAPE DUMP and LOAD commands. The Command Summary Log and the Volume Log are in the same format for SPXTAPE SCAN as they are for SPXTAPE DUMP and SPXTAPE LOAD. See Figure 21 on page 38 and Figure 22 on page 38 to see the format of the Command Summary Log and Volume Log, respectively.

```
spxtape scan b30 spool
SPXTAPE SCAN   INITIATED ON VDEV 0B30
Ready; T=0.01/0.01 11:56:11
SCANNING 0B30     :       244  FILES,  PAGES      6,642
SCANNING 0B30     :       436  FILES,  PAGES     15,380
. ...
. ...
SCANNING 0B30     :     2,615  FILES,  PAGES     69,025
SCANNING 0B30     :     2,847  FILES,  PAGES     77,557
. ...
. ...
SCANNING 0B30     :     3,084  FILES,  PAGES     86,444
SCANNING 0B30     :     3,245  FILES,  PAGES     95,191
. ...
. ...
SCANNING 0B30     :     4,311  FILES,  PAGES    130,732
SCANNING 0B30     :     4,453  FILES,  PAGES    139,457
SPXTAPE SCAN END-OF-TAPE ON VDEV 0B30;
MOUNT NEXT TAPE
TAPE NUMBER:          0B30-001
FILES PROCESSED:        4,531
SPOOL PAGES:          147,392
RDR FILE 0143 SENT FROM MAINT    CON WAS 0143 RECS 4542 CPY  001 T NOHOLD
```

Figure 19. CP SPXTAPE Commands, Scan Examples

Note that when SPXTAPE LOAD is used to restore files to a system, you need to load all the tapes used for the SPXTAPE dump set. A single user's file may be spread over many tapes, and SPXTAPE has no way of knowing the total file count dumped for a particular user, or system, or when all the tapes have been mounted. Therefore, SPXTAPE LOAD must be manually ended with the SPXTAPE END command. Once the SPXTAPE END command is issued, the Command Summary Log is generated. You should also be aware that there is no need to have the tapes in any particular order to restore files to the system with SPXTAPE LOAD.

```
spxtape scan b30 rdr
SPXTAPE SCAN   INITIATED ON VDEV 0B30
Ready; T=0.01/0.01 11:50:57
HCPSPO1847E Tape on virtual device 0B30 contains data that is not valid.
HCPSPO1847E The tape is positioned incorrectly or the volume is not
            in SPXTAPE format.
HCPSPO1847E Ending processing of this volume
SPXTAPE SCAN END-OF-TAPE ON VDEV 0B30 WITH ERRORS;
MOUNT NEXT TAPE
TAPE NUMBER:         0B30-001
FILES PROCESSED:            0
SPOOL PAGES:                0
RDR FILE 0139 SENT FROM MAINT    CON WAS 0139 RECS 0012 CPY  001 T NOHOLD
```

Figure 20. CP SPTAPE Format Tape is Scanned with SPXTAPE

2.2.6 Message Recording and Journaling

When using SPTAPE, the resulting console is dominated by SPTAPE messages, with one message for every file that is dumped, loaded, or scanned. This effectively ties up the screen and overloads the operations staff with messages.

The SPXTAPE approach to monitoring progress is to put the messages tracking the processed files into a log file, much like a console log. Each SPXTAPE command generates at least two log spool files, a Command Summary Log and one or more Volume Logs. SPXTAPE logs are created for each SPXTAPE DUMP, LOAD or SCAN command invoked. These files have a spool type of CONS, and are spooled to the same user regular console logs are spooled to. To ensure that the correct virtual machine receives the logs, an appropriate CP SPOOL CONS command should be issued before SPXTAPE processing is started.

A Command Summary Log is created for each logical SPXTAPE command (an explanation of a logical SPXTAPE command may be found in section "SPXTAPE DUMP and the Append Option" on page 34). This Command Summary Log indicates progress of the command and its status - for example, completed successfully, completed with errors or canceled.

An example of a Command Summary Log file is shown in Figure 21 on page 38.

```
03/04/94 15:26:42 MAINT    SPXTAPE DUMP B30 RDR MODE COMP
SPXTAPE DUMP   INITIATED ON VDEV 0B30
SPXTAPE DUMP END-OF-TAPE ON VDEV 0B30;

MOUNT NEXT TAPE

TAPE NUMBER:         0B30-001
FILES PROCESSED:        4,531
SPOOL PAGES:          147,401

SPXTAPE DUMP END-OF-TAPE ON VDEV 0B30;

MOUNT NEXT TAPE

TAPE NUMBER:         0B30-002
FILES PROCESSED:        3,385
SPOOL PAGES:          146,746

SPXTAPE DUMP COMPLETED  ON VDEV 0B30

TAPE NUMBER:         0B30-003
FILES PROCESSED:          182
SPOOL PAGES:            8,569

SPXTAPE DUMP COMMAND COMPLETED  ON VDEV 0B30

TIME STARTED:        15:26:42
TIME ENDED:          15:47:34

TAPE COUNT:               003
FILES PROCESSED:        8,098
SPOOL PAGES:          302,716
```

Figure 21. SPXTAPE Command Summary Log

A Volume Log is created for each tape volume. It lists files that are dumped, loaded, or scanned on that volume (tape or tape cartridge) with a similar format as the response from a CP QUERY RDR/PRT/PUN command, and includes additional information which answers the questions: where, when, what, who, and how much. If any files cannot be processed, error messages are contained within the Volume Log.

An example of a Volume Log is shown in Figure 22.

```
03/04/94 15:26:42 MAINT    SPXTAPE DUMP B30 RDR MODE COMP
SPXTAPE DUMP END-OF-TAPE ON VDEV 0B30;

MOUNT NEXT TAPE

TAPE NUMBER:         0B30-001
FILES PROCESSED:        4,531
SPOOL PAGES:          147,401

USERID   FILE QUEUE FILENAME FILETYPE OPENDATE OPENTIME ORIGINID SIZE
OP1      8647 RDR   REDPRINT NOTIFY   12/21/93 11:06:14 ITSCHELP 0000000
OP1      8561 RDR                     01/11/94 02:01:01 SYSADMIN 0000074
OP1      8648 RDR   NEWSCLIP F0940124 01/24/94 11:59:17 ENDY     0000000
OP1      8556 RDR   *UNKNOWN NOTICE   12/13/93 12:09:07 ISCTSTA  0000000
....
....
....
USERC    0116 RDR   REDPRINT NOTIFY   03/03/94 15:22:15 OP1      0000000
OP1      3594 RDR   WP60TUT  NFOBIN   01/12/94 22:13:36 NWERNER  0000007
```

Figure 22. SPXTAPE Volume Log

In addition to the log files, SPXTAPE generates a message approximately every 15 seconds, to show the progress that SPXTAPE is making. This feedback to the SPXTAPE operator is reassurance that all is well and progress is being made. These *heartbeat* messages from SPXTAPE are shown in Figure 23 on page 39. The heartbeat message contains a completion percentage. This is the percentage of spool pages already dumped, compared to the total number of spool pages SPXTAPE expects to be dumped.

```
spxtape dump b30 rdr mode comp
SPXTAPE DUMP   INITIATED ON VDEV 0B30
Ready; T=0.01/0.01 15:26:42
DUMPING  0B30     :        96  FILES,  PAGES      3,124    1% COMPLETE
DUMPING  0B30     :       237  FILES,  PAGES      6,078    2% COMPLETE
DUMPING  0B30     :       340  FILES,  PAGES      9,499    3% COMPLETE
DUMPING  0B30     :       426  FILES,  PAGES     12,764    4% COMPLETE
....
....
....
DUMPING  0B30     :     4,523  FILES,  PAGES    146,836   48% COMPLETE
SPXTAPE DUMP END-OF-TAPE ON VDEV 0B30;
MOUNT NEXT TAPE
TAPE NUMBER:         0B30-001
FILES PROCESSED:       4,531
SPOOL PAGES:         147,401
DUMPING  0B30     :     4,531  FILES,  PAGES    147,408   48% COMPLETE
DUMPING  0B30     :     4,598  FILES,  PAGES    156,150   51% COMPLETE
....
....
DUMPING  0B30     :     7,916  FILES,  PAGES    294,154
SPXTAPE DUMP COMPLETED  ON VDEV 0B30
TAPE NUMBER:         0B30-003
FILES PROCESSED:         182
SPOOL PAGES:           8,569
RDR FILE 0129 SENT FROM MAINT    CON WAS 0129 RECS 0190

SPXTAPE DUMP COMMAND COMPLETED  ON VDEV 0B30
TIME STARTED:        15:26:42
TIME ENDED:          15:47:34
TAPE COUNT:              003
FILES PROCESSED:       8,098
SPOOL PAGES:         302,716
RDR FILE 0126 SENT FROM MAINT    CON WAS 0126 RECS 0033
```

Figure 23. CP SPXTAPE Console Output

When the console spooling is enabled during a SPXTAPE operation, it is possible to send all output to a single user. In Figure 24 on page 40, a sample RDRLIST shows a spooled console, a Command Summary Log, and a Volume Log for three operations. Notice that the Volume Log contains the virtual device, dump set, date, time information, and SPXTAPE operation in the filename and filetype of the spool file. In this case, the CONSOLE was spooled class A.

```
 VMAINT   RDRLIST  A0  V 164  Trunc=164 Size=9 Line=1 Col=1 Alt=0
Cmd   Filename Filetype Class User  at Node     Hold  Records  Date      Time
      (none)   (none)   CON A VMAINT   TOTVM1   NONE      204 05/17    16:21:10
      0517DUMP 163253   CON A VMAINT   TOTVM1   NONE       27 05/17    16:32:53
      0517D181 16325301 CON A VMAINT   TOTVM1   NONE       81 05/17    16:32:59
      (none)   (none)   CON A VMAINT   TOTVM1   NONE       31 05/17    16:36:11
      0517SCAN 163642   CON A VMAINT   TOTVM1   NONE       19 05/17    16:36:42
      0517S181 16364201 CON A VMAINT   TOTVM1   NONE       70 05/17    16:38:29
      (none)   (none)   CON A VMAINT   TOTVM1   NONE       43 05/17    16:39:09
      0517DUMP 163922   CON A VMAINT   TOTVM1   NONE       27 05/17    16:39:22
      0517D181 16392201 CON A VMAINT   TOTVM1   NONE      107 05/17    16:41:09


1= Help      2= Refresh  3= Quit      4= Sort(type) 5= Sort(date) 6= Sort(user)
7= Backward  8= Forward  9= Receive  10=           11= Peek      12= Cursor
```

Figure 24. CP SPXTAPE Sample RDRLIST After Three Operations

2.2.7 SPXTAPE Performance

SPXTAPE attempts to load and dump files in a minimum amount of time by overlapping and optimizing the DASD and tape I/O activity. To do this, it uses system virtual storage to buffer and optimize I/O. SPXTAPE requires a minimum of 512 KB of virtual storage per tape drive used. SPXTAPE also allows multiple tape drives to be used, which contributes to throughput by reducing operator intervention, and allows for use of multiple CHPIDs to the tape drives.

How SPXTAPE Compares to SPTAPE

In testing, we dumped 8,000 files using SPTAPE and compared it to SPXTAPE using various configurations, as shown in Table 4. The system that was used was a second-level guest system with a single large spool area on a 3390 device. Hence, the spool environment was a bottleneck for this system. In testing performed on a system with four spool volumes and optimized for spool performance, tests using both SPTAPE and SPXTAPE to dump 34,000 files are shown in Table 5.

Command used	Number of drives	Mode	Time in minutes
SPTAPE	one	not available	66
SPXTAPE	one	nocomp	21
SPXTAPE	three	nocomp	22
SPXTAPE	one	comp	23
Note: The test environment was not optimized for performance, 8000 files.			

Table 4. CP SPXTAPE and SPTAPE Relative Performance

Command used	Number of drives	Mode	Time
SPTAPE	one	not available	3 hours 12 min.
SPXTAPE	one	nocomp	22 minutes
Note: The test environment was optimized for spool performance, 34000 files.			

Table 5. CP SPXTAPE and SPTAPE Relative Performance

Apparently, even in a guest environment where an I/O bottleneck exists, SPXTAPE offers more than a 300% improvement in throughput compared with

SPTAPE. In an environment where no such bottleneck exists, it has been shown that SPXTAPE performance is improved by more than 850% over SPTAPE.

SPXTAPE Testing Notes

From Table 4 on page 40, you may consider that it is not worth using more than one tape drive nor using MODE COMP. Using multiple drives in the test case lessened any delay with rewind/unload/mount times, although it resulted in a slightly longer time to dump the files. The practical application of three tape drives means less operator intervention, and if the tape drives and the spool volumes are connected to multiple CHPIDs, there is an increase in throughput over a single output device.

For the testing that was performed on the guest system, the spool files were not representative of a typical system as these files were multiple duplicate files for a small number of user IDs. SPTAPE took three 3490E cartridges to contain the 8,000 spool files, as did SPXTAPE in MODE NOCOMP. SPXTAPE in MODE COMP reduced tape usage to two cartridges.

On the system used to dump 34,000 spool files, SPXTAPE reduced the number of volumes required to seven compared to SPTAPE′s eleven cartridges.

Because of its high performance, SPXTAPE puts a significant load on the spooling subsystem and associated I/O. Each drive that is used to transfer files has a loading effect similar to approximately 64 to 128 active spool users. This loading of the spooling subsystem can be restrained by appropriate CP SHARE settings on the user ID that is performing SPXTAPE.

For further information about SPXTAPE, see *VM/ESA CP Command and Utility Reference*.

2.3 Share Capping and Proportional Distribution

To provide accurate control of resources within a VM/ESA system, the CP scheduler has been changed to provide a delivery of *surplus* processing in proportion to users′ share and to provide a very flexible limit on CPU resource allocated.

The benefits of having a limit on the ability of a user ID to use CPU is welcomed by installations that provide services based on CPU time used. For example, a user may enter into a program loop. If the user is being billed for the time used, both the user and the service provider are placed in a difficult position. With the ability to set limits on how much CPU resource that a user ID can use, VM/ESA Release 2.2 represents an improvement in usability.

The CP scheduler was also enhanced so that CP ABSOLUTE SHARE can be adjusted to tenths of a percent. Also, the default settings for SRM STORBUF have been changed. See section 2.10, “Miscellaneous Enhancements to CP” on page 68, for more information about the SRM STORBUF changes.

2.3.1 Maximum Share

Historically in VM/ESA, the scheduler allowed only specification of the minimum share to be allocated to a particular user ID. This resulted in users who have been able to exceed their entitlement and use additional system CPU resources with no way of proper control.

The scheduler now allows a way to control maximum share for each user. The target maximum share applies only to processor resources. The minimum share, or the guaranteed amount of share, has not been changed and still applies as it did previously - to processor, storage, and paging to DASD.

With properly set minimum share and maximum share values, you can guarantee the availability of resource, without over allocating when *surplus* processor resource is available.

2.3.2 Proportional Distribution

In previous releases of VM/ESA, when there were many users in the dispatch list, only a few of them used resources according to their share setting. Many users would be I/O bound, therefore not using system resources to which they were entitled; this resulted in *surplus* resources being available.

Historically, the scheduler algorithms have not directed the surplus in an explicit manner, so there has been no provision to control this resource. The surplus tended to go to the users with the largest share setting. This, in turn, has allowed these users to obtain more than what they would reasonably expect to receive. The interactive users, who normally have less share than, for example, a guest operating system, missed the opportunity to benefit from the surplus resources.

The CP dispatcher list priority calculation now allocates surplus processing among all the users who can use it, in proportion to their share settings.

2.3.3 Changed Commands and Directory SHARE Statement

The CP command SET SHARE and the CP directory SHARE statement now allow you to specify a second set of ABSOLUTE and RELATIVE share settings that control the maximum values. There are also new limit types used to control the dispatch of excess processor resources.

The following limit types are available:

NOLIMIT This is the default setting. There is no maximum share limit imposed. The effect is the same as pre-VM/ESA Release 2.2 systems.

LIMITSOFT The user does not receive more than the specified limit if there is any other user who can use the surplus without exceeding their own limit, if set. If no other user can use the surplus, the limit is overridden to give this user more than the maximum share specified. A user who has not reached their maximum share or has NOLIMIT set, has priority over LIMITSOFT users for surplus CPU resource as it is made available.

LIMITHARD The user does not receive more than the specified limit, even if no other user can use the surplus processing capacity. The surplus processing capacity is not used when all users have a LIMITHARD share setting and have reached their share.

The CP Command SET SHARE is changed to support the new limit setting parameters.

The format of the CP SET SHARE Command is shown in Figure 25.

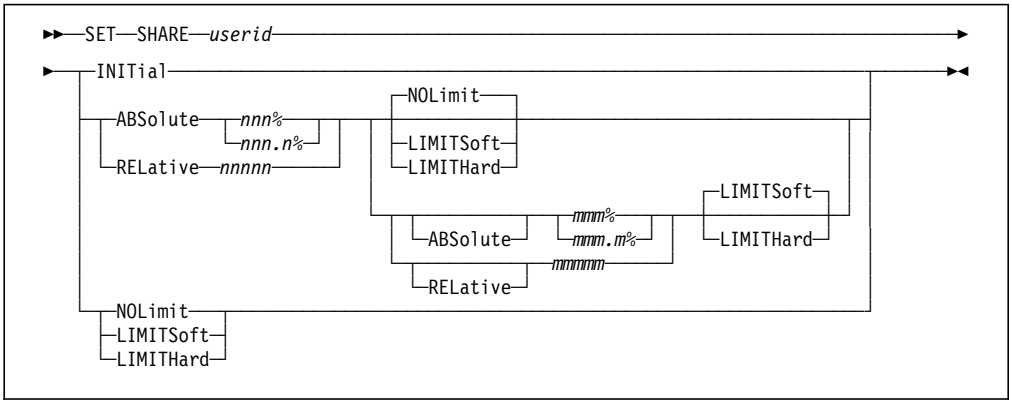


Figure 25. CP SET SHARE Command Syntax

For more information, see *VM/ESA CP Command and Utility Reference*.

The DIRECTORY SHARE statement allows new parameters to provide for the imposition of a limit on a user's share setting. The format of the Directory SHARE statement is shown in Figure 26.

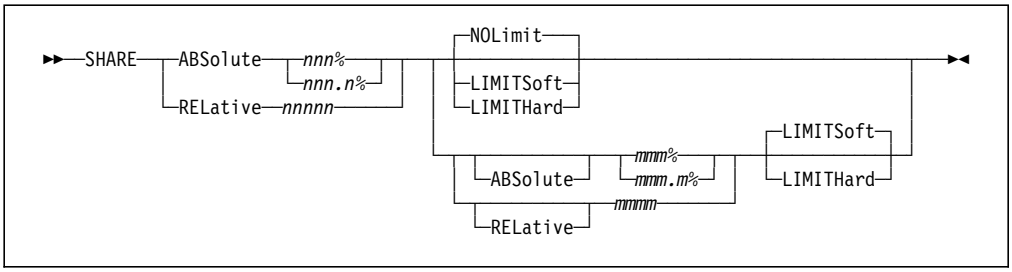


Figure 26. SHARE Directory Control Statement

For more information about CP SHARE, see *VM/ESA Planning and Administration*.

Absolute Share Decimal Setting

The absolute share setting for any user ID can be set to include tenths of one percent. This change is a result of the incremental increase in processor power over time requiring finer scheduler controls, and as a reaction to customer requests. As an example, one percent of a 9021-982 may be too much to give to a small guest system, so now the share can be further broken down using tenths of one percent increments.

Changed CP Command Responses

The responses from both the CP QUERY SHARE and CP SET SHARE commands reflect the limits imposed on a user ID. The response to the CP privilege class E versions of the CP INDICATE LOAD and CP INDICATE QUEUE commands have also been changed to include the number of users limited. CP INDICATE LOAD command shows the number of users in the *limit list*; that is, users who have reached their maximum share. The CP INDICATE QUEUE command indicates

individual users in the limit list. CP INDICATE QUEUE includes a L1, L2 or L3 list status if the user is in the limit list. These dispatch list designations have the same meaning as E1, E2, and E3, however the affected users have limited maximum share.

Note: Because the responses from SET SHARE and QUERY SHARE in VM/ESA Release 2.2 are different from previous releases, each installation should review locally developed applications that rely on output from either of the above commands.

The response from a class E CP INDICATE QUEUE is shown in Figure 27. The responses from various CP QUERY SHARE and CP SET SHARE statements are shown in Figure 28. The response from a class E CP INDICATE LOAD is shown in Figure 29.

```
ind q
PVM           Q3 PS  000376/000373 RSCS          Q1 PS  001186/001164
MAINT         Q1 R00 000874/000854 TIMOTHY       L2 R   000264/000264
Ready; T=0.01/0.01 19:19:41
```

Figure 27. Response from CP INDICATE QUEUE Commands (Class E Privilege)

```
q share timothy
15:23:25 USER TIMOTHY : RELATIVE SHARE= 100
15:23:25                MAXIMUM SHARE= NOLIMIT
Ready; T=0.01/0.01 15:23:25

set share timothy rel 100 abs 1.8% limithard
15:24:56 USER TIMOTHY : RELATIVE SHARE= 100
15:24:56                MAXIMUM SHARE= LIMITHARD ABSOLUTE 1.8%
Ready; T=0.01/0.01 15:24:56

q share timothy
15:25:20 USER TIMOTHY : RELATIVE SHARE= 100
15:25:20                MAXIMUM SHARE= LIMITHARD ABSOLUTE 1.8%
Ready; T=0.01/0.01 15:25:20

set share timothy rel 1 limits
15:27:02 USER TIMOTHY : RELATIVE SHARE= 1
15:27:02                MAXIMUM SHARE= LIMITSOFT RELATIVE 1
Ready; T=0.01/0.01 15:27:02

set share timothy initial
15:29:06 USER TIMOTHY : RELATIVE SHARE= 100
15:29:06                MAXIMUM SHARE= NOLIMIT
Ready; T=0.01/0.01 15:29:06
```

Figure 28. Responses from CP QUERY and SET SHARE Commands

```
ind load
AVGPROC-012% 01 AVGVEC-000% 01 VECTOR-00000
XSTORE-000005/SEC MIGRATE-0000/SEC
MDC READS-000081/SEC WRITES-000012/SEC HIT RATIO-090%
STORAGE-010% PAGING-0000/SEC STEAL-000%
Q0-00001(00000)                          DORMANT-00034
Q1-00002(00000)           E1-00000(00000)
Q2-00000(00000) EXPAN-001 E2-00000(00000)
Q3-00000(00000) EXPAN-001 E3-00000(00000)

PROC 0000-000% VEC-000%

LIMITED-00001
Ready; T=0.01/0.01 13:43:02
```

Figure 29. Response from CP INDICATE LOAD Commands (Class E Privilege)

2.3.4 Monitor Record Changes

The new maximum share setting is included in several monitor records of scheduler, monitor and user domains. The monitor's state sampling has been extended to include states where users are limited because they have reached their maximum share. This state sampling applies to both system sampling and user high-frequency sampling.

Further additions to the monitor data are the SET SRM XSTORE setting, the amount of storage that the CP scheduler requires and the amount of storage that the various CP scheduler classes use. The changed MONITOR records are shown in Appendix E, "Monitor and Accounting Record Changes" on page 157.

For further information on these and all the VM/ESA Monitor records, see the MONITOR LIST1403 file, which is located on MAINT's 194 minidisk.

2.3.5 Testing Results

A test was performed to determine the effectiveness of limiting share. A *looping user* condition was created by coding an infinite loop in a REXX EXEC. This user ID was assigned SHARE RELATIVE 100 and not limited. This user ID proceeded to use 90 percent plus of a processor in an unconstrained environment.

The first test was to impose a LIMITHARD of SHARE RELATIVE 1. The user ID's processor usage immediately dropped down from 90 percent plus, to below 1 percent of a uni-processor guest system. The second test was to change the limit to LIMITSOFT of SHARE RELATIVE 1. As no other user on the test system required CPU resource, the user ID again started to use CPU in the order of 90 percent.

A further test was done to see the effect of setting limits on a CMS user on a multiprocessor system. The results in various configurations are consistent with expectations. While the RELATIVE SHARE of the total system resource varies because of the effect of other users on the system, setting ABSOLUTE SHARE limits the user to a fairly consistent figure.

Imposing hard limits on suspected looping users would certainly be one way to ensure resources are not wasted. However, careful evaluation of each situation

is recommended, as productive work can be affected by improper use of share limits.

2.4 CLEAN Start Option and COLD Start Checkpoint Validation

CLEAN, an IPL option introduced into VM/ESA Release 2.2, initializes the spooling subsystem. No spool files whatsoever are recovered, including the System Data Files (SDF) - NSS, TRF, IMG, UCR, and NLS files.

In earlier releases of VM/ESA, COLD was the most regressive IPL option available. A COLD start would result in no standard spool files (RDR, PRT, or PUN) that were previously created being available after the IPL, but SDFs were still in the system spool.

Another improvement is the validation of the checkpoint area whenever a COLD start is requested. In order for the SDF component of the spool to be valid, the checkpoint and warm start data that is saved at shutdown must also be valid. If CP encounters invalid checkpoint data during IPL, it is possible that the warm start data, which is used to recover the SDFs, is also invalid. When this condition occurs during a COLD start, a message is displayed and the operator is given the choice of replying STOP, or to continue with the IPL by replying GO. If the reply is GO, CP attempts to recover as many of the SDFs as possible.

2.4.1 Using CLEAN Option During IPL

A time when the CLEAN option could be useful, would be when strings of DASD are migrated to new device types. The entire spooling subsystem data could be dumped using SPXTAPE, the system shut down, the devices replaced, IPL with a CLEAN start, and then restore all the spool with SPXTAPE.

When CLEAN is requested, a prompt is displayed requesting confirmation that this is the desired IPL option, much the same way as COLD requests confirmation. As with COLD, the operator responds with either *GO* to continue with the CLEAN start, or *STOP* to halt the procedure before any existing data is deleted. Unlike the COLD start option, there is no indication of how many spool files are deleted. During a CLEAN start, CP does not refer to any data that may exist in the warm start area, so it does not attempt to predict the number of files to be deleted. This also allows a CLEAN start to be very fast.

Prior to CLEAN start, the following procedure had to be followed to bypass warmstart or checkpoint problems:

1. Shutdown the system.
2. Format the warm start and checkpoint areas using a stand-alone program, such as ICKDSF.
3. COLD start the system.

The new CLEAN option at IPL time will be rarely used at most installations, but comes in handy when increasing the size of a checkpoint/warmstart area, creating a clone system, or merging two spool areas when backups are available.

If the operator chooses to stop processing, a X'9031' wait state is loaded.

An example of clean start is shown in Figure 30. The output required two screens of data.

```
 16:05:39 VM/ENTERPRISE SYSTEMS ARCHITECTURE   RELEASE 2.2 SERVICE LEVEL 9404;
 16:05:40 SYSTEM NUCLEUS CREATED ON 02/01/94 AT 11:13:07, LOADED FROM TOTE22
16:05:40
16:05:40 ************************************************************
16:05:40 * LICENSED MATERIALS - PROPERTY OF IBM*                      *
16:05:40 *                                                           *
16:05:40 * 5684-112 (C) COPYRIGHT IBM CORP. 1983, 1994. ALL RIGHTS   *
16:05:40 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,     *
16:05:40 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE   *
16:05:40 * CONTRACT WITH IBM CORP.                                    *
16:05:40 *                                                           *
16:05:40 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.           *
16:05:41 ************************************************************
16:05:41
 16:05:41 HCPZCO6718I Using parm disk 1 on volume TOTE22 (device ODB3).
 16:05:41 HCPZCO6718I Parm disk resides on cylinders 224 through 240.
 16:05:41
16:05:41 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
16:05:41       (NOAUTOlog)) or (SHUTDOWN)

6:06:36 CLEAN

                                                        MORE...   TOTVM1
```

```
16:06:36 NOW 16:06:36 EST THURSDAY 03/10/94
16:06:36 Change TOD clock (Yes|No)
16:07:03
16:07:03 The directory on volume TOTE22 at address ODB3 has been brought online
 16:07:03 HCPWRS2511A
 16:07:03 HCPWRS2511A CLEAN start has been selected.  This will cause all
 16:07:03 HCPWRS2511A spool files and System Data Files (NSS, DCSS, TRF, IMG,
 16:07:03 HCPWRS2511A UCR, NLS) to be deleted.
 16:07:03 HCPWRS2511A
 16:07:03 HCPWRS2511A No files have been deleted yet.
 16:07:03 HCPWRS2511A To continue CLEAN start and delete files, enter GO.
 16:07:03 HCPWRS2511A To stop CLEAN start without deleting files, enter STOP.
16:11:10 GO
 16:11:16 HCPWRS2512I Spooling initialization is complete.
16:11:18 DASD 0E60 dump unit CP IPL pages 1837
16:11:18 HCPAAU2700I System gateway TOTVM1 identified.
16:11:18 There is no logmsg data
16:11:18 FILES:   NO RDR,   NO PRT,   NO PUN
16:11:18 LOGON AT 16:11:18 EST THURSDAY 03/10/94
16:11:18 GRAF  0020 LOGON  AS  OPERATOR USERS = 1

                                                      HOLDING   TOTVM1
```

Figure 30. A Sample CLEAN START IPL

2.4.2 COLD Start Checkpoint Validation

The new validation of the checkpoint data provides an additional level of verification during COLD start processing. This gives the operator the chance to correct the problem before any invalid warm start data is read and SDFs are lost. This is an improvement in system management over previous releases. In the past, continuing with the COLD start may have resulted in meaningless messages, making it difficult to determine the real problem from the symptoms displayed.

The new checkpoint area validation is shown in Figure 31. If the checkpoint area eye-catcher, HCPCKPBK, does not exist where it should, CP prompts the operator. If the operator replies GO, CP attempts to read the warm start data and recovers SDFs. The existing COLD start confirmation prompt is also displayed.

```
18:25:42 HCPWRS9205A
18:25:42 HCPWRS9205A Checkpoint data is not valid.
18:25:42 HCPWRS9205A System Data file recovery data may not be valid.
18:25:42 HCPWRS9205A Continuation of the system IPL could result in the
18:25:42 HCPWRS9205A loss of system data files.
18:25:42 HCPWRS9205A
18:25:42 HCPWRS9205A To continue COLD start and attempt to
18:25:42 HCPWRS9205A recover system data files, enter GO.
18:25:42 HCPWRS9205A To stop processing, enter STOP.
```

Figure 31. COLD Start Checkpoint Validation Message and Prompt

2.4.3 EREP, Accounting, and Symptom Records

During IPL of a system, responding with the COLD or CLEAN start option results in the deletion of EREP, ACCOUNTing, and SYMPTOM records. These records are accumulated in a recording queue in storage. The RETRIEVE facility writes them from storage to a minidisk. It is possible for one or all of these types of records to accumulate in storage if RETRIEVE is not active.

When the system is shutdown, EREP, accounting, and symptom records that are in storage, but have not yet been retrieved to disk, are saved in the checkpoint area. WARM start restores the saved records to the recording queue in storage. FORCE start attempts to do the same as WARM start, but if it encounters any errors, FORCE start stops the recovery of these records and moves on to spool file recovery.

COLD start and CLEAN start do not attempt to recover EREP, accounting, or symptom records that may have been saved in the checkpoint area.

Migration Considerations

Please note that if you CLEAN start a VM/ESA 2.2 system, then revert to an earlier release, data that is deleted by the CLEAN start will not be restored.

2.5 Connectivity

This chapter describes several enhancements made to the connectivity part of CP. ISFC broadcast routing along with distributed IUCV provides new functions for easier use of a Communication Service (CS) collection.

2.5.1 ISFC Enhancements

The Inter System Facility for Communication (ISFC) is a function of CP that provides communication services between transaction programs. Using the services ISFC provides, such as APPC, transaction programs running on a VM system can communicate with programs that run on other VM systems or on workstations. These VM systems are called VM domain controllers. In a workstation environment, a programmable workstation running OS/2* or Novell**

Netware** can act as a domain controller workstation. These workstations must run VM Programmable Workstation Communication Services (VM PWSCS) and must be attached to the VM system. Controller workstations running AIX may also be defined in a CS collection, but they cannot be attached directly to VM systems.

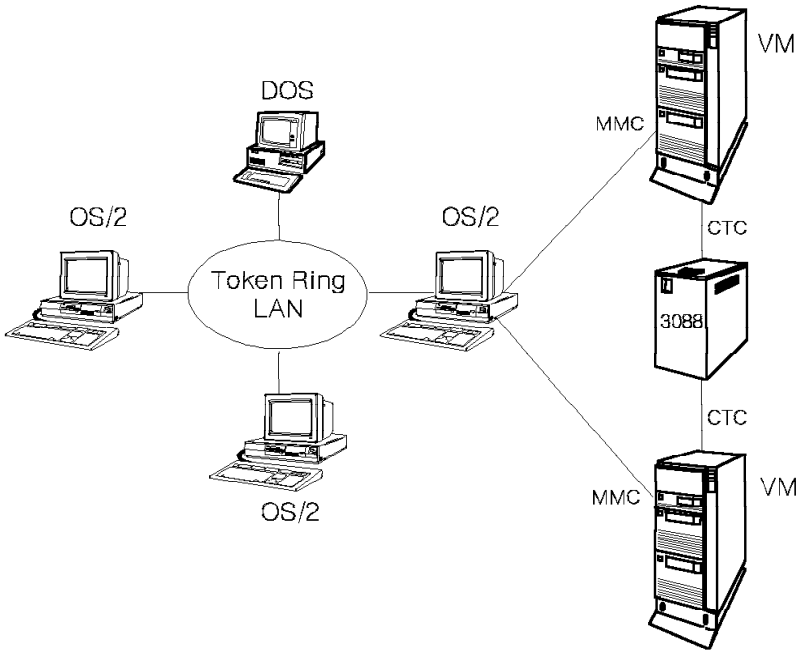


Figure 32. CS Collection with Several Domain Controllers

All ISFC domain controllers must be interconnected through channel-to-channel (CTC) paths to at least one other ISFC domain controller in a CS collection.

ISFC Broadcast Routing

The ISFC Broadcast Routing enhancement allows a source and target node in an ISFC collection that are not directly linked to each other to communicate.

This enhancement allows freedom in planning and hardware cost savings since all the VM nodes do not have to be directly connected to all other nodes in the collection. A node is able to access resources on the other node if the two nodes are directly linked, or if another path through the CS collection exists. A path exists if there are zero or more nodes between the source and target nodes. Instead of functioning as an end point, each node can forward incoming data to the rest of its neighbors or to a specific neighbor. This data includes new global resource identities and user ID location services.

Figure 33 on page 50 shows the potential reduction in CTC links when broadcast routing is used with six VM nodes.

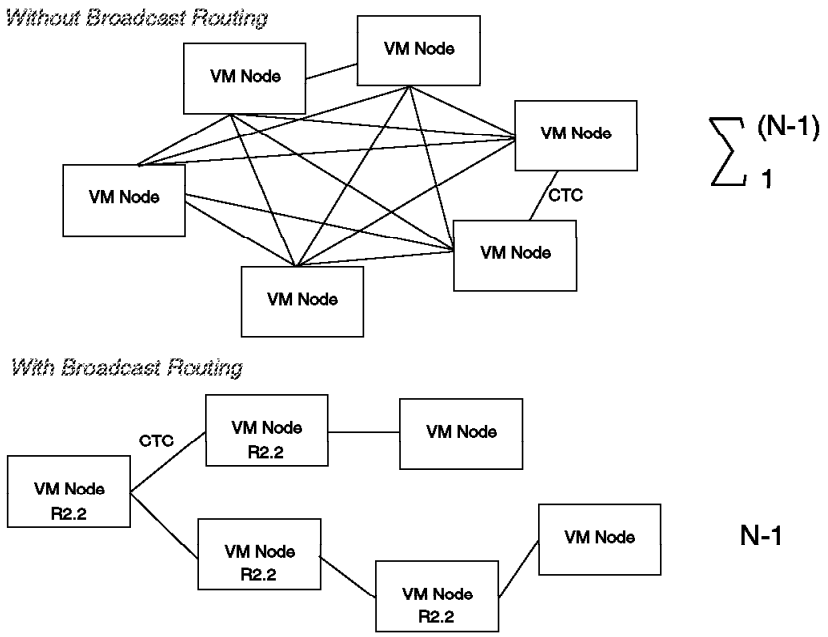


Figure 33. ISFC Broadcast Routing CTC Reductions

Only the VM nodes between source and target must have the new support to establish the path. The end nodes can be at earlier releases without broadcast routing.

ISFC broadcast routing support has the following features:

- Only an indirect path is necessary for full connectivity between source and target node.
- Global resources are automatically propagated throughout the collection.
- User ID directory broadcast within the collection for APPC requests requiring a user ID or password.
- Application conversation traffic management within the CS collection.

Before ISFC Broadcast Routing, the following points would be true about Figure 34 on page 51:

- Each node requires a direct CTC link to any node in the CS collection to communicate with each other.
- In the diagram, no communication exists between A and D and between B and C.

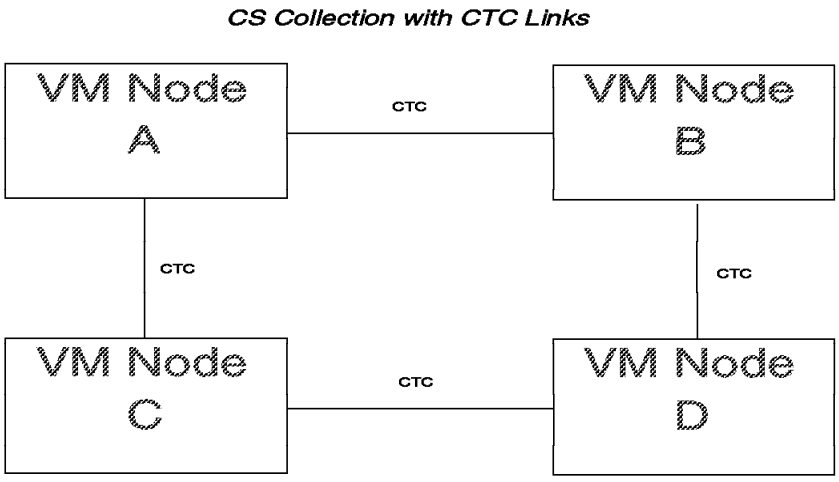


Figure 34. Before ISFC Broadcast Routing

In Figure 35, only three CTC links are needed to connect four VM nodes together. With the new support, every VM node in this collection can have access to all other nodes.

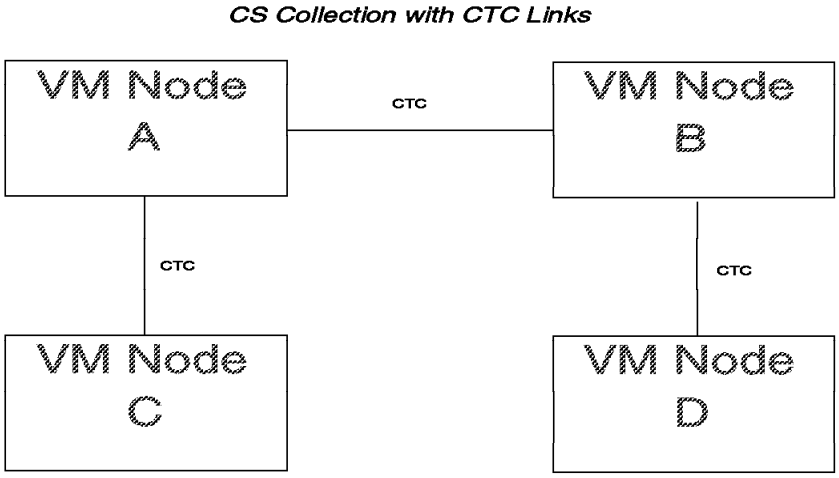


Figure 35. With ISFC Broadcast Routing

Transparent Services Access Facility and ISFC

A VM system that is running ISFC can also have a Transparent Services Access Facility (TSAF) virtual machine. If all systems in a TSAF collection support ISFC, TSAF may be removed and the systems may form a CS collection. The difference is that TSAF runs in a separate virtual machine and ISFC support is provided on the system by CP. This will improve the performance of communications between programs within the collection and may provide a more stable collection. ISFC greatly reduces administration overhead. Note that Coordinated Resource Recovery (CRR) is supported through ISFC, but not TSAF.

In addition, ISFC allows transaction programs on VM systems to have access to resources on workstations through VM PWSCS.

ISFC′s broadcast routing does not have all the function of TSAF′s full intermediate routing. TSAF can reroute communications previously active on a failed link if an alternate path is available; ISFC does not, and the communications server is required for error recovery. Also, various line drivers are still unique to TSAF, such as LAN and SNA. As it is not possible to couple a first level system to a second level system through a VCTC, TSAF is a solution in this case.

Migration Considerations

Now that fewer CTC connections are required, keep in mind a couple of points as you create larger CS collections:

- Resources, gateways, and user IDs with the same name on different nodes within a CS collection could create problems.
- Duplicate global resource names are not allowed.
- Beware of CS collections containing mixed VM releases. A back-level VM system has the same effect as an end node, and breaks the path between systems.

Backup Solution

If your systems have both direct and indirect ISFC routes, your communications can proceed over a indirect route when the direct route has failed. If more than one path exists between the source and target node, ISFC selects the most direct route when establishing a connectivity path. However, ISFC does not reroute an existing conversation if an improved path comes up.

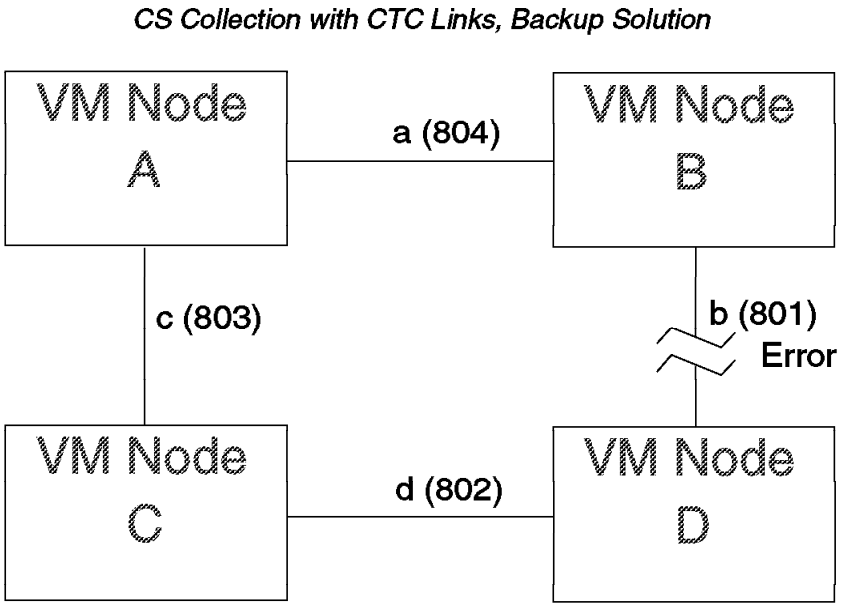


Figure 36. Backup Solution with ISFC Broadcast Routing

The discussion below pertains to Figure 36.

With the following commands, the CTCs and ISLINKs in the system are shown. All commands are issued from user ID STROHMAI, who is logged on at NODE-D. User ID STROHMAI wants to have access to file pool KIERAN, which is located at NODE-B. The user does not need to know were the file pool resides. He accesses the directory KIERAN:STROHMAI with filemode B and the subdirectory KIERAN:STROHMAI.WORK with filemode C. Since CTC 801 is the most direct route, communication would proceed through it (while the connection exists).

Now user STROHMAI copies a large file (84405 records) from his A-disk to the subdirectory KIERAN:STROHMAI.WORK, which is accessed with filemode B. During the copy operations, the CTC 801 was detached from system NODE-D. The following error messages were displayed:

```
DMSCPY105S Error 31 writing TEST FILE B2 on disk or directory

DMSJCA1152S File Pool KIERAN is unavailable; accessed directories for this file
filepool are released
```

Figure 37 shows the status of the links before and after the direct connection (CTC 801) was broken.

```
q ctc
CTCA 0801 ATTACHED TO SYSTEM  -ISFC
CTCA 0802 ATTACHED TO SYSTEM  -ISFC
Ready; T=0.01/0.01 14:33:17

q islink
Link: 0801              Type: CTCA
     Node:  NODE-B      Bytes Sent:           4601
     State: Up          Bytes Received:       4602
     Buffer Count:  16  Status: Idle
Link: 0802              Type: CTCA
     Node:  NODE-C      Bytes Sent:         468639
     State: Up          Bytes Received:    6198473
     Buffer Count:  16  Status: Idle
Ready; T=0.01/0.01 14:36:01

<<detach of CTC>>

q ctc
CTCA 0801 BOX/ATTC TO SYSTEM  -ISFC
CTCA 0802 ATTACHED TO SYSTEM  -ISFC
Ready; T=0.01/0.01 14:30:36
q islink
Link: 0801              Type: CTCA
     Node:              Bytes Sent:              0
     State: Down        Bytes Received:          0
     Buffer Count:  16  Status: Fatal I/O on link
Link: 0802              Type: CTCA
     Node:  NODE-C      Bytes Sent:         418843
     State: Up          Bytes Received:    6165222
     Buffer Count:  16  Status: Idle
Ready; T=0.01/0.01 14:03:30
```

Figure 37. ISFC Broken Connection

The file was not copied and the directories are released. When a DIRLIST command was issued, the directories were seen again. When the directories are accessed, the new connection is routed through CTC 802, 803, and 804 to NODE-B.

As another example, assume you are working in XEDIT mode at node D. You do some changes in a file, which resides on a subdirectory accessed with filemode B. Now, CTC 801 is detached from NODE-D. You now “FILE” out from the XEDIT session, attempting a return to DIRLIST in this case. The following error messages appear:

```
DMSJCA1152S File Pool KIERAN is unavailable; accessed directories for this file
pool are released

DMSWFL653E Error executing LISTFILE, rc=36
```

But, if you re-access the subdirectory, you will see the changes on the file are done through CTC 802, 803, and 804. The data integrity is maintained. Though ISFC didn′t restart the conversation, the availability of an indirect route caused the operation to complete once the new conversation, caused by the DIRLIST re-access, was established.

ISFC I/O Lock Removal and Queue Buffer Handler

VM/ESA Release 2.2 introduces an enhancement to the methodology used by ISFC to manage its buffer queues and also its I/O locking mechanism.

Previously, ISFC serialized all activity by using the ISFC global lock. This can create a bottleneck, especially on systems with a large amount of ISFC traffic. This enhancement provides the ISFC interfaces to CP′s I/O services and CTC line drivers with their own link lock. As a result, a lock that is activated on a CTC line driver does not adversely affect other ISFC communications facilities. This should improve throughput in a CS collection.

The queue buffer manager, a new feature of ISFC in VM/ESA Release 2.2, provides a general queueing mechanism for transferring data between two asynchronous CP processes.

In previous releases of VM/ESA, when ISFC needed to transfer data between two services, direct calls had to be made to CP. Now, when data needs to be transferred, ISFC queues a call to the queue buffer manager. The process that initiated the activity (the process that called the queue routine) may then continue processing, while the other service can complete asynchronously.

For additional information about ISFC communication, see *VM/ESA Connectivity Planning, Administration and Operation*.

2.5.2 Distributed IUCV

With VM/ESA 2.2, ISFC now allows Inter User Communication Vehicle (IUCV) programs to communicate with other IUCV programs within a CS collection. Distributed IUCV gives users the possibility to communicate with other users, regardless of where the partners physically reside, in the same manner as APPC/VM with ISFC links. Any user or server of a system can have access to servers of another system in a CS collection. This support provides data sharing

with IUCV across systems. This helps to reduce the number of duplicate servers. All systems in an ISFC collection at Release 2.2 and above can use distributed IUCV. To enable this support, you must add a new statement with options in the SYSTEM CONFIG file.

Figure 38 illustrates a basic distributed IUCV scenario.

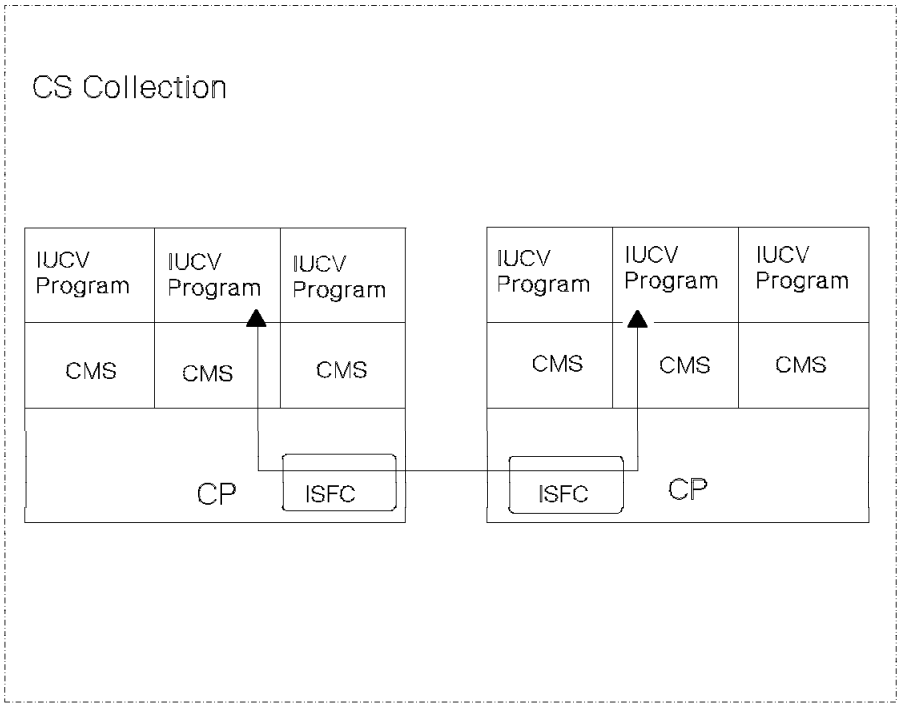


Figure 38. IUCV Communication in a CS Collection

SYSTEM CONFIG File

There is a new DISTRIBUTE statement in the SYSTEM CONFIG FILE, shown in Figure 39. It allows you to specify distribution features for the local system.

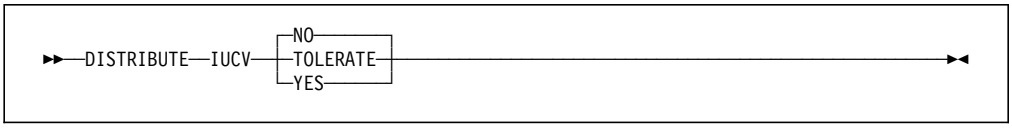


Figure 39. DISTRIBUTE Statement Syntax

One of the following options can be set:

NO Indicates that no distributed IUCV traffic is allowed to or from other nodes within the ISFC collection. This is the default.

TOLERATE Indicates that distributed IUCV is allowed to or from any other node in the ISFC collection. However, IUCV traffic will only leave the local node if the application specifies the target system name. The system name is specified on the TARGET parameter of the IUCV macro.

YES Indicates that this node participates fully in distributed IUCV across the entire ISFC collection. This means that target searches are done to the ISFC collection instead of the default of just the local system. A CONNECT request is attempted first on the local system; then, attempts are made to find the resource in the CS collection.

Though not the best example, in testing we were able to customize a SQL server not to use DRDA/APPC, then access it remotely using distributed IUCV. Other than the customization of the server, no application modification was required. Of course, since we used ISFC links to distribute the IUCV, you should use APPC to connect to a remote SQL server, not IUCV.

IUCV Macro

Programs on VM/ESA systems can be written using the IUCV assembler language programming interface. You can invoke all IUCV functions through the IUCV macro. In general, specify the name of the IUCV function you wish to perform, the address of the parameter list that contains input to the function, and the keyword parameters. The IUCV CONNECT function establishes an IUCV path to another virtual machine. You are not able to use this path until you receive an IUCV Connection Complete external interrupt for this path. This means the target has accepted the connection.

With the distributed IUCV enhancement, you have two new parameters on the IUCV CONNECT, shown in Figure 40.

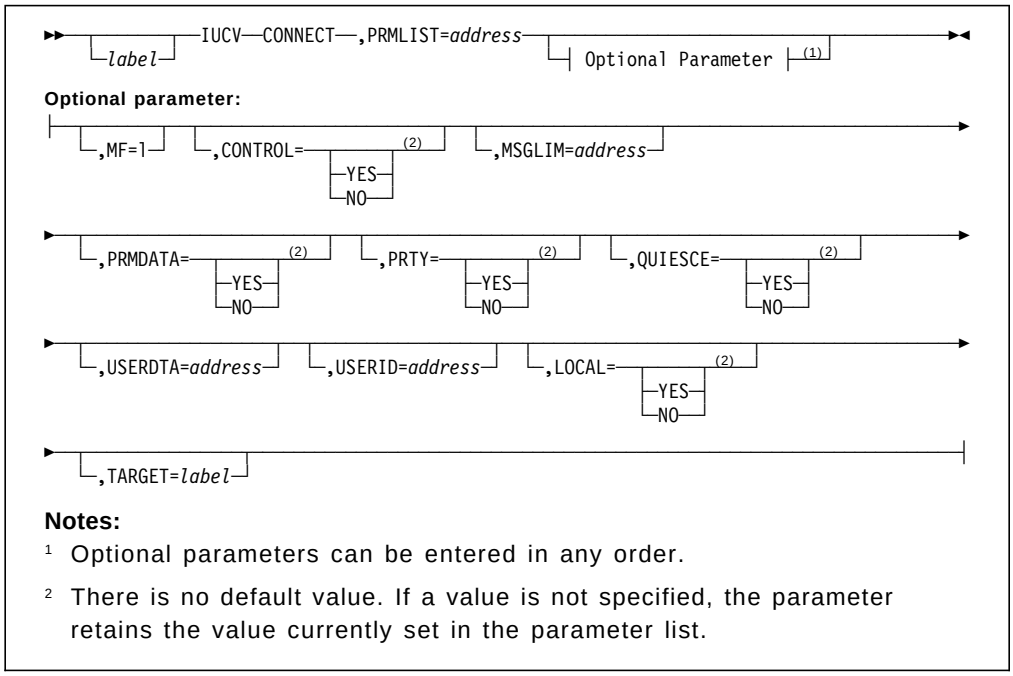


Figure 40. CP IUCV CONNECT Function Syntax

Explanation of the new optional parameters:

LOCAL= Allows an application to force the target to be on the local system. LOCAL=YES and TARGET=*label* cannot be specified together.

TARGET= Specifies which system the target must be on.

Applications

IUCV applications work the same in a distributed environment as they work on a local system with the following exceptions:

- PURGE and REJECT only are honored on the local system. Once a message is sent to other systems, it is considered delivered.
- The options PRIORITY and MSGLIMIT of the IUCV directory control statement must be present on both systems if they are to be honored.
- The maximum data length is limited to 1MB per message.

2.5.3 IUCV Virtual MP

The IUCV facility in Release 2.2 is enhanced to improve application performance in a virtual multiple processor (MP) environment. VMCF was already supported in virtual MP. In earlier VM releases, only the base processor could participate in IUCV or APPC/VM communications. Now, any processor in a virtual MP environment can run IUCV communications. For example, if a virtual machine is defined with three virtual processors (BASE, CPU02 and CPU03) and CMS is IPLed using CPU03, MP applications can issue IUCV functions on BASE and CPU02. With the new support, your application would be able to declare buffers on different virtual processors, enable the IUCV interrupts on the needed processor, handle the interrupts, and route them to the appropriate virtual processor.

Without appropriate guest operating system support, it is difficult or impossible to use IUCV in a virtual MP environment. IUCV virtual MP was designed for future usage of IUCV and APPC/VM. CMS and GCS do not currently support virtual MP functions.

How to Set Up Virtual MP

You can define up to 64 virtual CPUs with the MACHINE control statement of the user directory.

```
MACHINE ESA 64
```

Then you have two possible ways to bring up the processor configuration:

- If you want to define 3 CPUs and the user is not logged on, use the CPU control statement in the user directory.

  ```
  CPU 1 BASE
  CPU 2
  CPU 3
  ```

  Define a BASE processor only on one CPU statement. If BASE is not specified on any CPU statement, the CPU statement with the lowest address becomes the BASE.

- While running in a virtual machine, use multiple DEFINE CPU commands to initiate a virtual configuration.

  ```
  DEFINE CPU 1 BASE
  DEFINE CPU 2
  DEFINE CPU 3
  ```

With the QUERY command, you can determine whether the DEFINE CPU commands have been carried out.

QUERY V CPU

Enhanced IUCV Functions

In the following list, an overview of the enhanced IUCV functions is given.

- Processor specific functions:

  DECLARE BUFFER
  IUCV functions may be invoked by any virtual processor if one of the processors has issued a DCLBFR function. After the interrupt buffer is defined and the virtual processor enabled for IUCV interrupts, the virtual machine can now receive IUCV external interrupts. The difference between a virtual single processor and a virtual MP environment is, that the IUCV interrupts are treated as floating interrupts, presented on any available virtual processor that has an IUCV buffer declared.

  RETRIEVE BUFFER
  The IUCV RTRVBFR function only retrieves the buffer for the currently running virtual processor.

  SETMASK and SETCMASK
  The function of SETMASK and SETCMASK only applies to that processor on which they are invoked. This allows an application to direct different types of IUCV interrupts to different virtual processors.

- Virtual configuration specific functions:

  DESCRIBE, TESTCMPL, and IPOLL
  These functions were changed so they could complete on any virtual processor in a virtual MP environment.

  TESTMSG
  If multiple virtual processors issue the TESTMSG function, you will not know in which order the processors are taken out of wait state.

  QUERY
  In addition, CMS and GCS can use IUCV QUERY during initialization to determine the maximum length input parameter list extension (IPARMLX) that is accepted on an APPC/VM connect.

2.6 Logon BY

Logon BY is a new facility that allows one user ID, using its own password, to logon to another user ID.

Logon BY avoids the security exposure of sharing passwords for multiple VM service machines between many users. Some usual examples are the OPERATOR and MAINT virtual machines, where its typical for a whole department of people to know a single shared password.

To implement Logon BY, enhancements were made to the following:

- VM user directory
- LOGON command
- New CP QUERY BYUSER command
- Access Control Interface Parameter List (ACIPARMS)
- DIAGNOSE code X'260'
- DIAGNOSE code X'26C'
- Accounting records

When an External Security Manager (ESM), such as RACF*, is installed, the authorization provided by the LOGONBY statement might be completely replaced by some other authorization criteria given by the ESM. Refer to documentation provided by your ESM for further details.

A class G user can interrogate their own virtual machine's BYUSER ID using the CP QUERY BYUSER command, or DIAGNOSE X'260' subcode X'00000004'. Using DIAGNOSE X'26C' subcode X'00000004' or the CP QUERY BYUSER *userid* command, a class E user can additionally query someone else's BYUSER ID.

2.6.1 VM User Directory

A new LOGONBY statement in a user's directory entry defines the users authorized to logon and use this ID with their own passwords. The format of the new LOGONBY directory control statement is shown in Figure 41.

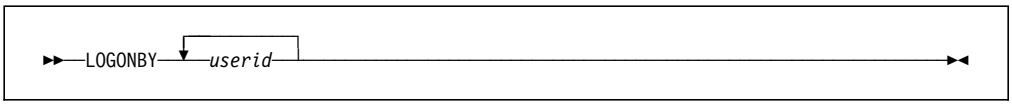


Figure 41. CP LOGONBY Directory Control Statement

where:

userid Specifies the user ID of a user who is authorized to enter the LOGON command with the BY operand for the virtual machine whose directory entry contains this statement.

A directory entry may contain multiple LOGONBY statements, specifying up to a total of eight user IDs. The LOGONBY statement must precede any device statements.

An example user directory entry containing the LOGONBY statement is shown in Figure 42.

```
USER USERA AAAAAA 64M 64M G
MACHINE ESA
IPL CMS
LOGONBY USERB USERC
CONSOLE 009 3215
⋮
MDISK 0191 3380 030 20 E22U01 MR

USER USERB BBBBBB 64M 64M G
MACHINE ESA
IPL CMS
CONSOLE 009 3215
⋮
MDISK 0191 3380 030 20 E22U54 MR

USER USERC CCCCCC 64M 64M G
MACHINE ESA
IPL CMS
CONSOLE 009 3215
⋮
MDISK 0191 3380 030 20 E22U27 MR
```

Figure 42. CP LOGONBY User Directory Entry

The example above shows the USERA directory entry. USERB and USERC can logon USERA with their own passwords. When USERB logs onto USERA, USERB's password is used for LOGON authorization checking. USERB is titled the BYUSER for USERA.

2.6.2 LOGON Command

A new BY operand on the LOGON command makes it possible to logon to another user ID using your user ID's password. The BY option on the LOGON command works together with the LOGONBY directory statement.

LOGON BY Examples

Referring to the example directory entries given in Figure 42 on page 59, some logon scenarios are documented below.

In Figure 43, a simple LOGON with the BY option is shown.

```
L USERA BY USERB
ENTER PASSWORD  (IT WILL NOT APPEAR WHEN TYPED):
BBBBBB <--- Password as entered, though not normally visible

There is no logmsg data
FILES:   NO RDR,   NO PRT,   NO PUN
LOGON AT 16:26:09 EST MONDAY 02/28/94
VM/ESA REL. 2.2 02/01/94 11:19

Ready; T=0.02/0.02 16:26:14
```

Figure 43. CP Logon BY Command

Another example where the BY and HERE parameters are used together on the LOGON command with the password is shown in Figure 44

```
L USERA CCCCCC BY USERC HERE
DISCONNECTED FROM GRAF  08E2
There is no logmsg data
FILES:   NO RDR,   NO PRT,   NO PUN
RECONNECTED AT 11:17:24 EST WEDNESDAY 03/02/94
B
```

Figure 44. CP Logon BY HERE Command

Notes:

1. The *password* (CCCCCC in Figure 42 on page 59) *must* be given after the user ID and before the BY operand. It does not work the other way around.
2. The PASSWORDS_ON_CMDs with the LOGON parameter YES in the FEATURES statement of the system config file was specified to allow the supplied password to be accepted.

The new LOGONBY directory control statement does not affect, in any way, the common LOGON procedure of the target user ID. The USERA virtual machine can still be logged on by itself, using its own password, as shown in Figure 45 on page 61.

```
L USERA
ENTER PASSWORD  (IT WILL NOT APPEAR WHEN TYPED):
AAAAAA <--- Password as entered, though not normally visible

There is no logmsg data
FILES:   NO RDR,   NO PRT,   NO PUN
LOGON AT 18:38:20 EST MONDAY 02/28/94
VM/ESA REL. 2.2 02/01/94 11:19

Ready; T=0.02/0.02 18:38:22
```

Figure 45. CP LOGON Command

LOGO Consideration

None of the examples shown in section 2.6.2, “LOGON Command” on page 60 used the LOGO panel interface. As with the HERE option of the LOGON command, the new BY option is supported on the “COMMAND ===>” field of the LOGO panel, the secondary panel obtainable by pressing enter from the LOGO panel, but not as an option on the “USERID ===>” field of the LOGO panel.

LBYONLY Password

Similar to the NOLOG password, a new reserved directory password is introduced with Logon BY: *LBYONLY*.

When LBYONLY is specified on the USER directory control statement, the user ID is only allowed to be logged using the BY option of the LOGON command. You cannot logon to this user using LBYONLY as the password. A user ID with the password LBYONLY is not useful on the LOGONBY directory statement of another user. You cannot have this user gain access to the other user ID using their LBYONLY password through the Logon BY command. LBYONLY can never be used as a logon password to gain access to a userid.

2.6.3 CP QUERY BYUSER Command

A new CP QUERY BYUSER user command queries and displays the BY user ID if one exists. A CP privilege class G user may query the user they are logged onto to determine what the BYUSER is. This could allow applications to customize profiles particularly for each user in a department as they access a shared userid. Class E extends the command to allow a user to query the BYUSER for every user ID on a system.

Referring to the examples given in Figure 43 and Figure 44 on page 60, and Figure 45, the corresponding QUERY BYUSER commands are shown in Figure 46 on page 62.

```
query byuser usera
The BYUSER for USERA is USERB
Ready; T=0.01/0.01 16:26:37

query byuser usera
The BYUSER for USERA is USERC
Ready; T=0.01/0.01 11:17:47

query byuser usera
There is currently no BYUSER for USERA
Ready; T=0.01/0.01 18:38:32
```

Figure 46. CP QUERY BYUSER Command

2.6.4 Logon BY Auditing and Accounting

The following sections discuss the details specific to Logon BY security, programming interfaces, and auditing.

ACIPARMS

The CP Access Control Interface piece of the *RPI system service (IUCV) has been enhanced, so the BYUSER value can be passed to the External Security Manager (ESM) when the ESM is called to authorize LOGONs.

The ACIPARMS parameter list for an HCPRPWEP call for the LOGON command has the following changes:

Label	Contents
ACIRGRP	Access control group name from the user directory entry of the virtual machine on which the logon is being done.
ACIRUSR	User ID of the virtual machine on which the logon is being done.
ACIFLAG2	ACINPASS is set if the user ID being logged on is defined with the NOPASS operand in its directory entry.
ACIBYVAL	The user ID following the BY operand of the LOGON command, if the BY operand was specified, or else zeros.

For a user to exploit this function on a system that does use an ESM, the user ID must meet the requirements defined by the ESM for this function. These requirements do not necessarily include the use of the Logon BY directory statement.

DIAGNOSE Code X′260′ and X′26C′

The DIAGNOSE Code X′260′ (Access Certain Virtual Machine Information) is not changed in the behavior or use of function subcode X′00000000′.

Function subcode X′00000004′ is new. It returns BYUSER information for the invoking virtual machine.

A new code added in VM/ESA Release 2.2 is the DIAGNOSE Code X′26C′ (Access Certain System Information). It has one function subcode, X′00000004′, that allows a class E user ID to determine the BYUSER of any system user ID.

For details about DIAGNOSE X′260′ or X′26C′, see *VM/ESA CP Programming Services*.

Accounting Records

BYUSER information is added to the Type 4 Accounting record for LOGON journaling.

A BYUSER is a user who attempts to logon to a virtual machine using the BY operand of the LOGON command. In this case, the BYUSER′s password is used for LOGON authorization checking for the virtual machine. So, the invalid password attempts are counted against the BYUSER ID, not the user ID that is the target of the LOGON.

For more information on the Type 4 Accounting record, refer to section E.4, “Accounting Record Type 4” on page 158.

2.7 Asynchronous Data Mover Support

The Asynchronous Data Mover Feature (ADMF) hardware support provides an extension to the existing channel subsystem that enables the moving of paging activity for certain functions to the I/O processor. This frees the instruction processor for other work.

ADMF was initially announced for ES/9000 711-based and 511-based processors for DB2 applications running under MVS/ESA*. ADMF efficiently moves large amounts of data between central and expanded storage. Up to a 20 percent reduction in elapsed time for selected DB2 queries can be expected when using a suitably configured system.

Support for the Asynchronous Data Mover is introduced in VM/ESA Release 2.2. This support is limited to providing the ability on an ADMF capable processor for an appropriate level of MVS/ESA and DB2. The MVS guest system must be running as an ESA mode preferred guest. Such a guest system may be referred to as a qualified guest.

ADMF is made available for a qualified guest that:

- Has VM/ESA Release 2.2 running natively on the hardware
- Has the Dynamic Relocation Facility (DRF) hardware feature available
- Is a preferred guest (V=R, V=F) of VM.

At system initialization, VM/ESA CP configures the ADMF and prepares it for use. As a qualified guest logs on, it is automatically allocated subchannels. These subchannels are dedicated to the guest system′s configuration for as long as it is logged on.

When this qualified guest logs off, the previously allocated ADMF subchannels are removed. The error handling for ADMF subchannels is the responsibility of the guest system.

Full guest recovery support for V=R systems is supported by VM/ESA Release 2.2 for ADMF. The presence of ADMF is confirmed through the CP Monitor. For Monitor records that have changed, see Appendix E, “Monitor and Accounting Record Changes” on page 157.

When IOASSIST is enabled, ADMF performance in a VM/ESA guest system is comparable to native ADMF performance.

2.8 Working Allegiance (MVS Sysplex DASD Sharing)

One of VM′s strengths is its ability to share DASD among many second-level guests. Among MVS guests, there is a new Cross-System Coupling Facility (XCF) that puts new challenges on VM′s DASD-sharing.

As part of VM′s virtualization of a guest′s channel programs, it is often necessary to simulate some CCW commands and execute other CCW commands on the real device. Some CCW commands can be executed on the real device, perhaps in a modified form, while also being partially simulated. The result is that a guest′s channel program is often executed on the real device in segments; the guest′s virtual channel program becomes two or more real channel programs.

For DASD being shared among guests using minidisks, this means that the virtual-channel-program segments from one guest can be interleaved with segments from other guests. In non-VM environments, where real DASD are being shared among real processor-complexes, no such interleaving of channel programs is possible. (For technical background information, see “Working Allegiance” in “Enterprise Systems Architecture/390 Principles of Operation.”)

XCF relies on working allegiance for the particular locking protocols implemented through its channel programs for the MVS couple dataset. Prior to VM/ESA Release 2.2, APAR VM51012 alleviated problems due to the lack of a virtual working allegiance for VM′s minidisks. Beginning with Release 2.2, VM/ESA provides a new command, SET WRKALLEG (Set Working Allegiance) and corresponding directory options to provide a fully-simulated working allegiance for selected minidisks.

To avoid potential database corruption in a MVS XCF environment under VM, the SET WRKALLEG command or its corresponding directory options must be used for the minidisk containing the MVS couple dataset.

Please note that the SET WRKALLEG command and associated directory options provide working allegiance simulation only within a single VM/ESA system. Thus, a MVS sysplex cannot extend beyond a single VM/ESA system, and no such support has been announced by IBM.

This potential MVS sysplex exposure is detailed in the following external sources, available through your account team:

- *Washington Systems Center FLASH 9313*
- *Informational APAR II06583*
- *VM/ESA APAR VM51012*.

2.8.1 Implementation

The enhancements described below provide simulated working allegiance for specified minidisks, intended for use with minidisks containing MVS XCF couple datasets. A new option, called WRKALLEG, on the MINIOPT and DASDOPT statements and a class G SET WRKALLEG command can be used to request a virtual working allegiance while guest channel programs are running. A class G QUERY WRKALLEG command is provided to determine the current setting of the virtual working-allegiance function for a specified virtual DASD.

Virtual working allegiance is completely transparent to guests. It is conceptually similar to VM′s virtual reserve/release support for minidisks but is not controlled by guest CCW commands and lasts only for the duration of each single virtual channel program.

Channel programs from guests with read-only access to a minidisk that is protected by virtual working allegiance can still be segmented and interleaved with channel programs from other read-only guests. Any channel program from a guest with write access, however, does not begin until all ongoing channel programs from read-only-access guests have completed (for example, all segments have completed). Once a channel program from a guest with write access has started, no other channel programs from any guest is started until all segments of the active guest channel program have completed.

Virtual working allegiance should not be used for minidisks other than those containing a MVS XCF couple dataset, since the serialization described above unnecessarily affects guest′s I/O performance on minidisks that do not require it.

2.8.2 Directory Settings

Figure 47 shows the syntax of the MINIOPT directory statement with NOWRKALLEG and WRKALLEG, which follows a MDISK statement defining a minidisk that does not comprise the entire real DASD. NOWRKALLEG is the default.

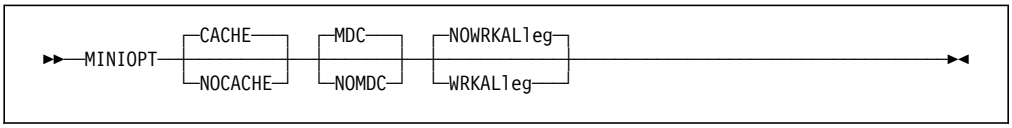
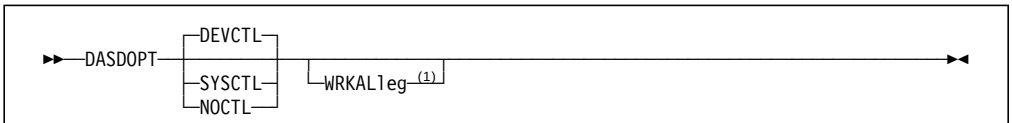


Figure 47. MINIOPT Directory Control Statement

Figure 48 shows the syntax of the DASDOPT directory statement, which follows a MDISK statement defining a full-pack minidisk. NOWRKALLEG does not exist for this statement, only explicit use of the WRKALLEG parameter activates working allegiance on the DASDOPT directory statement.



Note:

[1] The WRKALLEG parameter does not apply when the DASDOPT statement follows a DEDICATE or LINK directory statement.

Figure 48. DASDOPT Directory Control Statement

2.8.3 CP Commands for Working Allegiance

The CP privilege class G user is allowed control of the working allegiance environment. This allows a PROFILE EXEC for a MVS guest to tailor the virtual machine environment without the aid of a system administrator.

The SET WRKALLEG command, shown in Figure 49, allows individual setting for selected devices. Only one MVS guest in a guest-sysplex needs to issue SET WRKALLEG ON for a given minidisk.

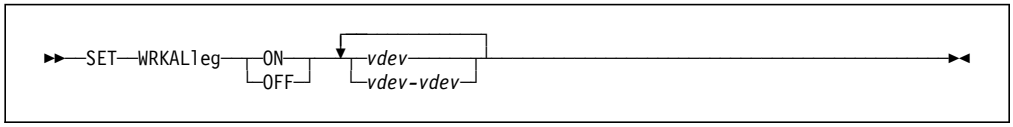


Figure 49. CP SET WRKALLEG Command

The following conditions apply to the SET WRKALLEG command:

- The command fails when:
  - Disk is in R/O mode
  - Device is dedicated
  - SHARED=YES is set for device.
- Virtual working allegiance does not apply to active guest I/O that was started before the completion of the SET WRKALLEG command.

The QUERY WRKALLEG command is a general user class method of determining the working allegiance status of the virtual devices.

The command response is synchronous, which would allow trapping the response in an EXEC. This command is shown in Figure 50.

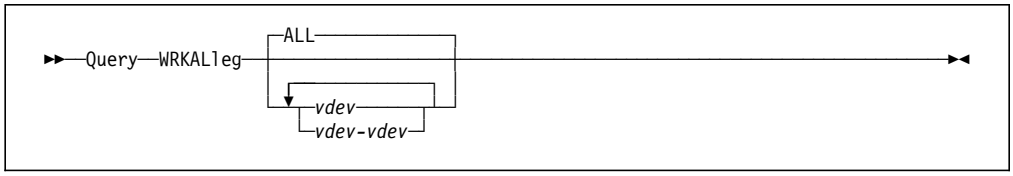


Figure 50. CP QUERY WRKALLEG Command

2.8.4 Examples of Working Allegiance Command Responses

The following example shows the output of the CP commands discussed in the previous section.

```
set emsg on
Ready; T=0.01/0.01 18:45:01
q wrkal
Working allegiance is not simulated for any minidisk in your configuration.
Ready; T=0.01/0.01 18:45:10
q wrkal 100
HCPWAC040E Device 0100 does not exist
Ready(00040); T=0.01/0.01 18:45:26
q wrkal 191
DASD 0191 working allegiance is turned off.
Ready; T=0.01/0.01 18:45:30
set wrkalle on 191
DASD 0191 working allegiance is being simulated among same system guests.
Ready; T=0.01/0.01 18:46:02
link * 191 391 rr
DASD 0391 LINKED R/O; R/W BY VETTER
Ready; T=0.01/0.01 18:46:23
q wrkal 391
HCPWAC201OE DASD 0391 is read only, WRKALLEG is rejected.
Ready(02010); T=0.01/0.01 18:46:32
q wrkal
Working allegiance is being simulated for the following minidisks:
0191 0391
Ready; T=0.01/0.01 18:56:23
```

Figure 51. Working Allegiance Command Examples

2.9 CICS Subspace Support for MVS

As a continuing effort for VM to support the latest hardware features, VM/ESA Release 2.2 introduces guest support for the Subspace-Group Facility (SGF). The Subspace-Group Facility was designed to allow CICS* to isolate transactions from themselves. This means CICS is running in the base address space and its transactions are running in their own subspace. It is intended that each subspace contain a different subset of the storage in the base space, thus achieving isolation between the transactions so they cannot damage each other's data.

The VM/ESA Subspace-Group Facility support is available for guest system usage only. Second level guests can be either XA or ESA mode machines; it is not provided for 370- or XC-mode guests.

The Subspace-Group Facility consists of a new machine instruction, Branch in Subspace Group (BSG), and modifications to five existing instructions. To allow the new support, a simulation routine was added to simulate the BSG instruction. This new support is mostly object code only (OCO).

For debugging of assembler programs, the TRACE, DISPLAY, and DUMP commands were enhanced.

2.10 Miscellaneous Enhancements to CP

This section briefly describes some other enhancements made to the CP component of VM/ESA Release 2.2.

Changed Default Setting for CP SET SRM STORBUF

One of the first things that a systems programmer used to change on any VM system was the CP SET SRM STORBUF parameters. These default parameters were acknowledged as suitable for very few customers, so they have been changed. The default SRM STORBUF setting is now 125 105 95. It is likely to suit more customers than the old setting of 100 85 75.

New Options in the SYSTEM CONFIG File

Several new options have been added to the SYSTEM CONFIG file in VM/ESA Release 2.2. Full details on these options can be found in *VM/ESA Planning and Administration*. These options are briefly described here.

- CPADDON_INITIALIZATION_ROUTINES

  Used to generate a list of entry points in CP that are called during system initialization.

- DISTRIBUTE

  Specifies the features for the distributed IUCV. See section 2.5.2, "Distributed IUCV" on page 54 for further information on this option.

- INT_MITIME

  Used for setting the interval at which a device is checked for missing interrupts.

CP Support for Dynamic Time Zone Changes

VM/ESA Release 2.0 introduced the concept of changing the time zone at a VM installation without having to re-IPL CP. With VM/ESA Release 2.2, diagnose code X′274′ has been enhanced.

Diagnose X′274′ allows a program to request notification from CP whenever the time zone of the system was changed. This diagnose has a new option for a virtual machine′s control program, such as CMS, to request that CP send the virtual machine a time zone change interrupt.

CP now keeps track of two flags for generating time zone interrupts.

- The normal time zone interrupt flag to be used by application programs within the virtual machine.
- The new time zone interrupt flag for use by the operating system within the virtual machine.

If either or these two flags are set on, CP generates an external interrupt X′2004′ to the virtual machine. To enable this external interrupt, the virtual machine must set bit 19 of control register zero (CR 0) to 1.

For further information on this diagnose code, refer to *VM/ESA CP Programming Services*. Also, refer to section 3.4, "CMS Support for Dynamic Time Zone Changes" on page 82, for a description on how CMS uses this enhancement to dynamically change its time zone information.

Minor Change to CP INDICATE USER SYSTEM Command

The response from the CP INDICATE USER SYSTEM command now reflects a *NPREF=nnnn* value. This is the number of shared address space pages owned by the system that are residing on paging space. See *VM/ESA CP Command and Utility Reference* for a complete explanation of the INDICATE command.

Monitor-change Addressability to DCSS

Since VM/ESA Release 1.0, it has been possible to have more than one user ID connected to the *MONITOR CP system service. This is called the monitor multiplexor.

In previous releases of VM/ESA, the CP Monitor obtains addressability to the Monitor DCSS through the user ID that has the Monitor DCSS loaded. This causes serialization problems when the user ID that the CP Monitor is using for translation logs off the system. In VM/ESA Release 2.2, the CP Monitor recognizes that the user ID is logging off and attempts to find another user ID for translation. The CP Monitor now obtains addressability to the Monitor DCSS by attaching it to the system.

Modifications to CP Monitor internals, and storage management routines in particular, may be affected by the change to CP Monitor addressability. For example, a modification that causes the CP Monitor to produce a non-IBM defined monitor record may need to be rewritten.

Fast Path Locking Improvements

The logic used to generate a fast-path lock has been enhanced. This provides a modest performance improvement in the area of processor usage of mainline IUCV/APPC functions.

Integrated Console Support

VM/ESA Release 2.2 allows the operator console of your ES/9000 mainframe to be used, within limits, as the system console for your VM system. Please refer to section 8.1, “Integrated Console Support” on page 135 for further information on this enhancement.

Chapter 3. CMS Enhancements

This chapter describes the enhancements made to CMS in VM/ESA Release 2.2. The CMS in VM/ESA Release 2.2 is the last release of CMS to run in a 370 mode virtual machine. Users with a requirement for 370 mode CMS support are encouraged to run their 370 mode applications using the 370 accommodation facility, which was made available in VM/ESA Release 2.1.

370 mode virtual machines will continue to be provided for use by guest operating systems and by non-CMS based applications. Users of 370 mode CMS applications are encouraged to test their applications to ensure they run in XA/XC mode machines in 370 accommodation mode.

For further information on the CP SET 370ACCOM MODE command, refer to *VM/ESA CP Command and Utility Reference*.

3.1 VMLINK

At a brief look, VMLINK can be defined as the combination of the CP LINK and CMS ACCESS command. But this is not all true. On the CP LINK command, you need to provide a free virtual address. VMLINK does it for you. What about the CMS ACCESS command? In this case, you have to provide a free filemode. Here too, VMLINK resolves it for you. VMLINK can be defined as an easy way for VM support personnel and end users to link and access minidisks and Shared File System directories where products or applications reside. It eliminates the set of EXECs that LINK and ACCESS minidisks or directories, and can be used to start the application. VMLINK provides even more, solving the problems of product or application dependencies and offering a detailed view of the products and applications, with their levels, available on a system.

A more accurate definition of VMLINK is an integrated interface for:

- **Support Personnel**
  - VMLINK can be used to toggle between different levels of applications and program products without affecting the system users. This is particularly important while planning and performing a high impact product or application migration.
  - VMLINK acts as a bootstrap into many system functions and provides a common interface for applications, thus reducing complexity and support costs.
  - VMLINK maintains minidisk addresses and Shared File System directories in a VMLINK names file by using standard CMS nicknames managed by an enhanced CMS NAMES command.
- **Program and Applications**
  - Products and applications do not need to have their own interfaces to link and access minidisks and directories.
  - When source information changes, a single update can be made to the VMLINK names file, instead of changing this information within the application.

- **Users**
  - Users can choose the AUTOLINK option which provides a way to link and access disks or SFS directories at log on. This saves them from changing their PROFILE EXEC and enables system changes transparent to the user.
  - Products and applications that have files on multiple disks can have all required disks linked and accessed with a single command.
  - Users can easily locate, link, and access disks where products and applications are stored and identify saved segments and the user IDs that own the products.
  - Users cannot only easily determine what products are available on their system by scanning a list provided by the VMLINK panel interface, but also their levels.
  - Products and applications can be categorized so that a user can identify quickly what functions are available on the system to perform a particular task. For example, if the command VMLINK GRAPHICS is issued, the VMLINK panel displays a list of tools categorized as graphics tools.
  - Users can invoke a product or application using VMLINK by using the INVOKE option. This reduces the link, access, and invocation actions to a single command.
  - When invoking VMLINK for a product or application, product and application co-requisites, located on separate disks, can be linked and accessed by VMLINK. This can be done automatically, without any user knowledge about the co-requisites and interdependencies between products and applications.
  - If a disk is not identified in the VMLINK names file, VMLINK can still be used to link and access a disk. This is done by supplying VMLINK with the user ID and address, instead of the nickname. As explained before, VMLINK searches for a free address and a free filemode, saving the user′s time.

3.1.1 VMLINK Set Up

This section describes the necessary activities that must be followed by system support personnel after a product or an application installation.

There are two primary files that can be customized for additional tailoring of VMLINK:

- **VMLINK CONTROL**

  A control file for setting some VMLINK defaults, called the VMLINK CONTROL file. For more information about VMLINK CONTROL, refer to “VMLINK CONTROL File” on page 73, and appendix F.1.2, “VMLINK Control File and Names File” on page 168.

- **VMLINK NAMES file**

  A NAMES file that contains all the necessary information about the products and applications residing on the system. This information includes the virtual machines, where they reside, their minidisks, and directories, for example.

A system programmer can create a VMLINK NAMES file for system use. A user can maintain their own names file that adds or replaces values in the system VMLINK NAMES file. The default name for the user VMLINK NAMES file is USERPROD NAMES, but the name can be changed. For more information about the VMLINK NAMES file (defaults, filenames and override), refer to appendix F.1.2, "VMLINK Control File and Names File" on page 168, and sections "VMLINK CONTROL File" and "VMLINK NAMES Files" on page 172.

For creation and maintenance of these names files, the enhanced CMS NAMES command is used with the new option VMLINK.

Notes:

1. VMLINK CONTROL and VMLINK NAMES files are not shipped with the VM/ESA Release 2.2 base tapes. The system programmer should review their system requirements and determine if they are required to further customize VMLINK.
2. If an External Security Manager (ESM), such as RACF/VM, is installed on the system, system support personnel will need to update their ESM's authorization database to allow VMLINK to work correctly.

VMLINK has two additional areas which can be customized:

- **PROFVMLK XEDIT**

  This macro comes with the VM/ESA base tapes and is used by VMLINK when it is invoked without operands. Using the DEFAULTS command, it can be replaced with a customized version that changes the VMLINK default panel interface. For more information about this macro, refer to section "PROFVMLK XEDIT" on page 184.

- **VMLINK Exits**

  VMLINK provides three different exits at three different points of the VMLINK cycle. Two of them can execute EXECs, while the third one can execute commands, macros, EXECs, and modules. It is system support personnel's responsibility to code and activate the exits to fit them to their system requirements. Refer to section "VMLINK Exits" on page 184 for more details.

VMLINK CONTROL File

The VMLINK control file, VMLINK CONTROL, sets some defaults for VMLINK, including:

- The search ranges for free file modes
- The search range for free virtual device numbers
- The NAMES files to be searched
- NAMES files to add to the beginning of the default search order
- VMLINK behavior when it cannot access a disk.

The control file, written and maintained by system support personnel, is called the system control file. Also, users can code their own VMLINK CONTROL control files. These are called local control files.

VMLINK scans all the VMLINK CONTROL files in A-to-Z file mode order, but only the first instance of each type of control record is used. This means that a user

can override some of the defaults defined on the system VMLINK CONTROL with their own local VMLINK CONTROL residing on their A-mode minidisk.

VMLINK Defaults Override: The different control files require a hierarchy, since several files control similar functions. Here are the rules for overrides:

- A *local VMLINK CONTROL* control file overrides the *system VMLINK CONTROL* file.
- Defaults set in a *nickname entry* (see section 3.1.1, “VMLINK Set Up” on page 72) override defaults set in *local* and *system VMLINK CONTROL* control files.
- Operands specified with the VMLINK command override defaults set in a nickname entry.

VMLINK CONTROL File Example: Figure 52 shows a control file. Unconventionally, the asterisks are not comments, but part of the control file verbs.

```
*ADDFILE LCLFILE

*EQUATE VMXA1 WTSCXA  SYDVMXA
*EQUATE ESA22 RACNEA1 RACNEA2

*ERROR IGNORE

*EXIT EXTCTL

*FILES USERPROD NAMES * PRODUCT .NO * VMLINK NAMES S

*ID USERID

*MODES G-L

*PEXIT  PRXTCTL

*VDEV BFO
```

Figure 52. VMLINK CONTROL Example File

For a detailed VMLINK CONTROL control file example and description, refer to section “VMLINK CONTROL File Example” on page 171.

NAMES Files for VMLINK

As the CMS names file is a good repository to collect information about other users, it was thought that it would also be a good repository for information about the products and applications residing on the system. In the same way that a nickname is assigned to a user, a nickname can also be assigned for linking and accessing the minidisks or Shared File System directories where products or applications reside.

In the same way that you define a list of names for users, you can define a list of products or applications in your VMLINK names file.

On the other hand, the CMS NAMES command, with its panel interface, offers a very friendly way to handle the information in the NAMES file, as can be seen in Figure 53.

```
 ====> NAMES (Vmlink panel)    File: USERPROD NAMES    S2              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: APLD12X  Description: Linking Application 12X.
Product Linking    :
Information (USERID:
cuu/.DIR dirname)  :
Category: APLIDS
                   Invoke: MODULE BOOT12X
                  Preexit: PRVERIF .CU1
                     Exit: PSVERIF .FM1
              Valid Nodes: RACNEA2 WTSCXA
                         :
            List of Names: AP12X191 AP12X195 APLD13X
                         :
                         :
                         :
 Tag:               Value:
 Tag:               Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 53. A VMLINK NAMES Panel Interface

For a detailed VMLINK NAMES file example and description, refer to section "VMLINK NAMES Files" on page 172.

3.1.2 VMLINK Usage

VMLINK is a complete and integrated interface between system support personnel, programs or applications, and users, offering a wide range of flexibility for its use.

In Figure 54, a simple VMLINK command can be seen, linking and accessing one minidisk. It is important to note that only the owner user ID of the minidisk, **STROHMAI**, and its virtual address, **191**, were given. It was not necessary to specify a virtual device address to link the minidisk or a free filemode to access it.

```
Ready; T=0.01/0.01 13:00:01
vmlink strohmai 191
DMSVML2060I STROHMAI 191 linked as 900 file mode W
Ready; T=0.06/0.07 13:00:14
sp console stop close
```

Figure 54. VMLINK as a Simple Link and Access Command

VMLINK just took the information from the VMLINK CONTROL file to assign a virtual address and a free filemode.

Another, quite simple, example is just issuing VMLINK without parameters to use its panel interface, as shown in Figure 55.

```
 VERONICA VMLINK   AO  V 260  Trunc=260 Size=162 Line=1 Col=1 Alt=0
Cmd   Nickname Vdev Fm Ext Lm  Category  Description
      APL2     =    = /        ALL       APL Version 2
      CSP      =    = /        ALL       CSP 3.2.2
      EREP     =    = /        ALL       EREP error recording system
      ISPF     =    = /        ALL       ISPF/PDF development tool
      ISPF2    =    = /        ALL       High storage version of ISPF
      MSEMHV   =    = /        ALL       Mid Hudson Valley Education Courses
      GDDM     +300 * /        ALL       GDDM Version 3.1
      GDDM2    +301 * /        ALL       GDDM/XA version 2.3
      GDQF     +302 * /        ALL       GDQF version 2 release 2.
      DMS      +200 * /        ALL       DMS Version 2 Release 1
      DITTO    +300 * /        ALL       DITTO Version 3 5688-052
      PMF      =    = /        ALL       Page Management Facility
      PS370    =    = /        ALL       PostScript 370 interpreter
      OFFICE   =    = /        ALL       OfficeVision/VM Rel 2
      PROLOG   =    = /        ALL       Prolog AI programming language
      QMF      =    = /        ALL       Query Management Faciltiy
      QMS      =    = /        ALL       Quality Management System
1= Help       2= Refresh  3= Quit      4= Sort(name)  5= Link        6= Alt PF
7= Backward   8= Forward  9= Category 10= Detach     11= Filelist   12= Cursor
```

Figure 55. VMLINK as a Simple Link and Access Command

In Appendix F, “VMLINK Setup, Exit Activation and Using Examples” on page 159, the VMLINK topic is covered in much greater detail.

3.2 CMS Pipelines Enhancements

CMS Pipelines in VM/ESA Release 2.2 provides several new stages and enhancements to many existing stages. Information on how each stage command ends is explained in the reference documentation. Additionally, related help for CMS Pipelines is now provided as part of the VM/ESA help facility, giving you ready access to related stage command information.

3.2.1 Why use Pipelines?

CMS Pipelines is used by many customers to improve their productivity. CMS Pipelines lets you run programs concurrently, automatically passing data from one program to the next. Studies show that when CMS Pipelines, instead of a standard REXX EXEC, is used to solve a typical programming problem, it saves 15 to 300 percent in coding time. This percentage varies depending upon the task and the techniques applied to accomplishing that task. There are now 154 built-in stage commands documented in VM/ESA CMS Pipelines to help you solve most any task. Pipeline programs are coded in a modular design that encourages code reuse, flexibility, and helps to provide an object-oriented approach to programming.

3.2.2 New and Improved Stage Commands

CMS Pipelines provides the following new stage commands:

3270BFRA	FRTARGET	TOTARGET
3270ENC	GATE	XRANGE
APLDECODE	LDRTBLS	
APLENCODE	LISTPDS	
CRC	PAUSE	

CMS Pipelines improved the following stage commands:

BUILDSCR	LOCATE	SUBCOM
CASEI	NLOCATE	TAKE
CMS	RUNPIPE	XLATE
COMMAND	SNAKE	ZONE
CP	SPECS	
DROP	SPLIT	

For further information about CMS Pipelines built-in functions and how to write and debug your own stage command, see the following manuals:

- *VM/ESA CMS Pipelines Reference*, SC24-5592
- *VM/ESA CMS Pipelines User's Guide*, SC24-5609
- *CMS Pipelines Tutorial.*, GG66-3158
- *VM/ESA REXX User's Guide*, SC24-5465
- *VM/ESA REXX Reference*, SC24-5466.
- *VM/ESA Diagnosis Guide*, LY24-5250.

A brief description of what each new stage command does follows:

3270BFRA Use this stage command to convert a 2-byte unsigned integer to the 12-bit buffer address, or back again. This 12-bit address is required for many 3270 display devices. You could use 3270BFRA, for example, to write a text string to a specified line on a 3270 terminal.

3270ENC This stage command prepares a 64-character translate table used to convert binary values in the range B'000000' to B'111111' (0 to 63 in decimal) to displayable 1-byte graphic characters to be placed in a 3270 data stream. Like the 3270BFRA stage, 3270ENC would be used in displaying data on a 3270 terminal. This stage command must be the first stage of a pipeline.

For further information regarding 3270 data streams, see *3270 Information Display: Data Stream Programmer's Reference*, GA23-0059.

APLDECODE This stage command works in the same way as the CMS commands SET TEXT ON or SET APL ON. APLDECODE converts graphic escape characters into a single character based on the translate table specified.

APLENCODE This stage command is the converse of APLDECODE. APLENCODE takes values from one of two translate tables, APL or TEXT, and converts them to a 2-byte graphic escape sequence.

CRC The CRC stage command provides a Cyclical Redundancy Check (CRC) that calculates a checksum on its input stream records. The resulting checksum is a 16-bit binary number. CRC allows you to verify that a file has been correctly transmitted by comparing the checksum created prior to the transmission to the checksum created after the transmission. If the two CRC values do not match, the file is corrupted.

FRTARGET FRTARGET (an abbreviation of FROMTARGET) enables you to select all records starting with the first record selected by a specified stage command. FRTARGET invokes another stage

command and rejects all records until that specified stage command selects a record. For example, if “VSM” was the target, FRTARGET would allow you to create a list of VTAM controlled terminals from the CP QUERY NAMES command by rejecting all the user IDs up to the VSM user ID.

GATE The GATE stage command ends portions of a pipeline. For example, you could use GATE to look for a certain character string in a file, and when the character string is found, the data stops flowing through the pipeline; the GATE is closed for the data.

LDRTBLS Use LDRTBLS to run a compiled REXX user-written stage command that has been loaded into your virtual storage with the CMS LOAD command. It is possible to pass an argument to the routine from this stage command.

LISTPDS LISTPDS reads the directory of a CMS simulated partitioned data set, such as a MACLIB (macro library) or a TXTLIB (text library). After discarding the dictionary header record, one record is written for each member of the library. LISTPDS can only be used only as the first stage of a pipeline.

PAUSE Use the PAUSE stage command in a pipeline issued by RUNPIPE EVENTS to send a signal to the pipeline containing the RUNPIPE stage to receive a PAUSE event record. Use of this stage command could help synchronize or debug pipelines.

TOTARGET Like its companion stage FRTARGET, TOTARGET is used to choose specific records from an input stream. The TOTARGET stage command selects all records up to, but not including, the first record selected by a specified stage command.

XRANGE XRANGE creates one record containing a specified range of characters. XRANGE works similar to the REXX XRANGE function. For example, you could use XRANGE to build a set of hexadecimal DASD addresses to query on a system.

Enhancements to stage commands in VM/ESA Release 2.2 are described next.

- Six stage commands, CASEI, LOCATE, NLOCATE, SPECS, XLATE, and ZONE now contain the additional operands WORDSEPARATOR (WS), FIELDSEPARATOR (FS), WORDS, and FIELDS to specify input ranges to be processed. For instance, you can locate the third word in a record regardless of the column location.
- There is a change to the PIPE command, CALLPIPE and ADDPIPE pipeline subcommands, and the RUNPIPE stage command. You can now specify decimal, binary, or hexadecimal numbers on the MSGlevel operand.

A brief description of other enhanced stage commands follows:

BUILDSCR This stage command builds 3270 data streams from records in print-type format. For example, BUILDSCR could be used to view a file which was created in print format for a line printer.

The new operands are:

3278, 3279, 1
These values specify that the device on which the data streams will be displayed supports 3278 APL/TEXT

characters that follow a graphics escape character X′08′. The data streams may contain such characters.

3277, 2
These values specify that the device on which the data streams will be displayed supports 3277 APL/TEXT characters that follow a X′1D′ escape character. The data streams may contain such characters.

TEXT
This specifies that the two translate tables created correspond to the translations done when the CMS command SET TEXT ON is issued. This is the default.

APL
This specifies that the two translate tables created correspond to the translations done when the CMS command SET APL ON is issued.

CMS This stage command issues CMS commands from within a pipeline and writes the responses from the commands to the pipeline rather than to the terminal. The CMS stage command now allows the CMS return code to be written to the secondary output stream.

COMMAND Like the CMS stage command, COMMAND is used to issue CMS commands from within a pipeline and to write the responses from the commands to the pipeline rather than to the terminal. COMMAND now writes the return code to the secondary output stream.

CP Like the CMS and COMMAND stages, the CP stage issues CP commands from within a pipeline and writes the CP command responses to the pipeline rather than to the terminal. The CP stage command now writes the return code to the secondary output stream.

DROP Use the DROP stage to discard one of more records at the beginning or end of its primary input stream. DROP now provides an additional operand, *, that is used to specify that all records are written to the secondary output stream, if it is connected. If no secondary output stream is connected, all records are discarded.

RUNPIPE RUNPIPE is used to issue pipelines and to intercept the CMS Pipeline messages those pipelines produce as they run. The new EVENTS operand of RUNPIPE writes detailed information about the pipeline and its run progress, covering all pipelines issued by RUNPIPE. This information is formatted for use by another program. Examples of an event are dispatcher service, console I/O, and pause.

SPLIT The SPLIT stage command splits records into multiple records. Now an additional operand, MINIMUM, specifies the number of characters skipped before SPLIT begins searching for the target.

SUBCOM The SUBCOM stage is used to pass subcommands to a specified subcommand environment without intercepting the terminal output. Like the CMS, COMMAND, and CP stages, SUBCOM now provides the return code from the specified environment and writes it to the secondary output stream.

TAKE The TAKE stage command selects one or more records from the beginning or end of its primary data stream. It provides a new operand, *, which specifies that all records are written to its primary output stream. If the primary output stream is not connected, all records are discarded.

XLATE The XLATE stage is used to replace characters according to a translate table. The following code pages are added in VM/ESA Release 2.2:

437(ASCII) PC Display: U.S., Switzerland, Austria, Germany, France, Italy, U.K.

819(ASCII) ISO 8859 Latin Character Set 1 (Western Europe)

850(ASCII) PC Data-190: Latin Alphabet Number 1; Latin-1 countries.

863(ASCII) PC Display: Canada

865(ASCII) PC Display: Denmark, Norway

3.3 SYSTEM SEGID Enhancement

Previously, the addresses of logical saved segments were read from the SYSTEM SEGID file on the system disk (MDISK 190) during IPL of CMS. When a logical segment is re-saved (with the SEGGEN command), a new copy of SYSTEM SEGID is created with updated addresses. This new file must be copied to the system disk. But users cannot access the new logical segment addresses until CMS is re-IPLed. Attempts to use the new re-saved logical segments may result in abends or program checks, because the addresses that are being used are not correct. If you copy the SYSTEM SEGID file on the MDISK 190, you have to re-save the CMS segment to get a new shared S-disk directory. This could result in many pending-purge CMS segments, which exist in the system until all users of these segments have re-IPLed CMS or logged off.

In Release 2.2, the dependency on the logical segment addresses stored in the SYSTEM SEGID file is removed. Now, you do not need to copy the SYSTEM SEGID file to the system disk when only logical segment addresses have been changed. As a consequence, you do not need to re-save CMS, and there is no need to re-IPL CMS to get access to the new logical addresses.

You have to place the new copy of SYSTEM SEGID on the System disk only when one of the following events occur:

- New logical and/or physical segments have been created
- Existing logical and/or physical segments have been deleted
- The relationship between physical and logical segments has changed.

3.3.1 Physical and Logical Segments

Throughout the next section, examples of changes to the SYSTEM SEGID file are shown. Please note that only the first example requires no update to the MAINT 190 (S) disk, providing you are running VM/ESA Release 2.2.

Example of changing the addresses and length of logical segments

Current SYSTEM SEGID file:

```
PSEGMENT  CMSFILES  00700000  00100000  02/26/94  10:24:04
LSEGMENT  DMSDAC    00700000  0007CC40
LSEGMENT  DMSSAC    0077CC80  0006BB60
PSEGMENT  USER      00800000  00100000  02/26/94  16:32:39
LSEGMENT  USERDISK  00800000  0005BD40
LSEGMENT  USERSEG   0085BD80  00080020
```

After the SEGGEN command:

```
PSEGMENT  CMSFILES  00700000  00100000  02/28/94  12:23:18
LSEGMENT  DMSDAC    00700000  0007CC40
LSEGMENT  DMSSAC    0077CC80  0006BB60
PSEGMENT  USER      00800000  00100000  02/28/94  15:30:39
LSEGMENT  USERDISK  00800000  0007FFF0
LSEGMENT  USERSEG   00880030  0006CCC0
```

The above changes to SYSTEM SEGID file do not require updating the MAINT 190 System disk.

Example of creating new logical or physical segments

Current SYSTEM SEGID file:

```
PSEGMENT  CMSFILES  01450000  001B0000  02/01/94  12:24:08
LSEGMENT  DMSDAC    01450000  000CBD70
LSEGMENT  DMSSAC    0151BDB0  000C3D08
```

After the SEGGEN command:

```
PSEGMENT  CMSFILES  01450000  001B0000  02/01/94  12:24:08
LSEGMENT  DMSDAC    01450000  000CBD70
LSEGMENT  DMSSAC    0151BDB0  000C3D08
PSEGMENT  USER      00C00000  00100000  02/28/94  14:18:39
LSEGMENT  USERDISK  00C00000  0003B4D0
LSEGMENT  USERSEG   00C3B510  00078C80
```

The SYSTEM SEGID file must be copied to the system disk because USER, USERDISK, and USERSEG have been added.

Example of deleting existing logical or physical segments

Current SYSTEM SEGID file:

```
PSEGMENT  CMSFILES  01450000  001B0000  02/01/94  12:24:08
LSEGMENT  DMSDAC    01450000  000CBD70
LSEGMENT  DMSSAC    0151BDB0  000C3D08
PSEGMENT  USER      00C00000  00100000  02/28/94  14:18:39
LSEGMENT  USERDISK  00C00000  0003B4D0
LSEGMENT  USERSEG   00C3B510  00078C80
```

After the SEGGEN command:

```
PSEGMENT  CMSFILES  01450000  001B0000  02/01/94  12:24:08
LSEGMENT  DMSDAC    01450000  000CBD70
LSEGMENT  DMSSAC    0151BDB0  000C3D08
PSEGMENT  USER      00C00000  00100000  02/28/94  14:18:39
LSEGMENT  USERDISK  00C00000  0003B4D0
```

It is required to copy the SYSTEM SEGID file to system disk, because logical segment USERSEG no longer exists.

Example of changing the relationship between physical and logical segments

Current SYSTEM SEGID file:

```
PSEGMENT  CMSFILES  00700000  00100000  02/01/94  12:24:08
LSEGMENT  DMSDAC    00700000  0007CC40
LSEGMENT  DMSSAC    0077CC80  00020D60
PSEGMENT  USER      00800000  00100000  02/28/94  14:18:39
LSEGMENT  USERDISK  00800000  0005BD40
LSEGMENT  USERSEG   0085BD80  00080020
```

After the SEGGEN command:

```
PSEGMENT  CMSFILES  00700000  00100000  02/01/94  12:24:08
LSEGMENT  DMSDAC    00700000  0007CC40
PSEGMENT  USER      00800000  00100000  02/28/94  14:18:39
LSEGMENT  DMSSAC    00800000  00020D60
LSEGMENT  USERSEG   00820DA0  0005BD40
LSEGMENT  USERDISK  0087CB20  00080020
```

The SYSTEM SEGID file must be copied to the system disk because DMSSAC is now located in physical segment USER. Note that the changed addresses do not require that the SYSTEM SEGID file be copied.

3.4 CMS Support for Dynamic Time Zone Changes

VM/ESA Release 2.0 introduced the concept of changing the time zone on a VM installation without having to re-IPL CP. The Group Control System (GCS) changes to support this function were provided in VM/ESA Release 2.1.

Currently, CMS stores the following fields within the CMS nucleus at IPL:

- Current date
- TOD clock at midnight
- Time zone offset.

These values are not updated until the next IPL. If a time zone change occurs dynamically (using the function provided in VM/ESA Release 2.0), CP and CMS may get out of synchronization. CMS may incorrectly date stamp files and CMS applications may use the incorrect time. This line item corrects the problem.

We have already described the changes made to CP to allow for the generation of the interrupt to the CMS system running in the virtual machine (see page 68). CMS uses the external interrupt X′2004′ to determine when a time zone change has occurred. The fields within the CMS nucleus are not updated. This is to maintain compatibility with applications that may use these fields. The CVTTZ field of the CVT macro is updated to the current time zone by the CMS interrupt

handler when the external interrupt is received. In addition, the CMS DOS GETTIME support will reflect the time zone changes.

The CMS file system, Shared File System, EXECIO, and REXX functions (such as lineout, and time()) will all automatically reflect a CP SET TIMEZONE.

As REXX time() function is aware of this new interrupt, suppose a CMS machine has the following EXEC running:

```
/* Sample Rexx EXEC to Detect Time Zone Changes */
  address command
  do forever
     say time()
  end
exit(86)
```

In prior releases, if the operator entered SET TIMEZONE XXX to change the system offset, the REXX EXEC running in the CMS machine would not reflect the time change unless CMS was re-IPLed and the EXEC restarted. With the addition of CMS support for dynamic time zone, the CMS re-IPL is not required. The CMS dynamic time zone support reduces the number of CMS machines that require an IPL during a CP time zone change.

CMS Application Multitasking has been changed to support dynamic time zone, as follows:

- CSL routine *TimerStartTOD* has been enhanced such that timers set to expire at a certain local time will stay set to that local time, even if a zone change occurs after the timer is set.
- CMS signals a new event, *VMTIMECHANGE*, when the zone changes. This gives CMS applications a method to detect zone changes.

3.5 CMS Enhancements for Additional CP Spool Classes

CMS commands and utilities have become more flexible. Historical restrictions that CMS notes and other “mail” use CP spool class A have been lifted. You can now choose any punch CLASS from “A” through “Z,” “0” through “9” or an equal sign (=). If you specify an equal sign (=), the current punch class is used when sending a file. In this section, we review the changes required to support the new classes.

3.5.1 CMS SENDFILE Command

In previous releases of VM/ESA, the SENDFILE command always uses spool punch class “A” when sending files. There was no method for the user to select another class. This limitation causes several problems, such as when sending files from VM to MVS systems. Class A on a MVS system does not mean the same thing as class A on the VM system. The CLASS option is added to provide the capability for the user to specify which spool class to use when sending a file and to establish a class other than *A*, the default.

To send a file (test script a) to user MAINT with class C, you can enter at the command line:

sendfile test script a maint (cl c

If you want to use the SENDFILE MENU, enter SENDFILE at the command line with no options or parameters. There is a new field, the last one, which allows you to designate a new CLASS. In Figure 56 on page 84, we have changed the class to C.

```
                ----------------  SENDFILE  ---------------

File(s) to be sent    (use * for Filename, Filetype and/or Filemode
                           to select from a list of files)
Enter filename :
      filetype :
      filemode :

Send files to  :

Type over 1 for YES or 0 for NO to change the options:

   0    Request acknowledgement when the file has been received?
   1    Make a log entry when the file has been sent?
   1    Display the file name when the file has been sent?
   0    This file is actually a list of files to be sent?

   C    Spool class to use when sending the file(s)


   1= Help            3= Quit             5= Send               12= Cursor

====>
                                                          Macro-read 1 File
```

Figure 56. SENDFILE Menu with New CLASS Option

3.5.2 CMS NOTE Command

To send a note with class C to another user, enter at the command line:

note veronica (cl c

Figure 57 shows the header of the note with the changed class.

```
 STROHMAI NOTE     A0  V 132  Trunc=132 Size=9 Line=9 Col=1 Alt=0

* * * Top of File * * *
OPTIONS: NOACK    LOG    SHORT     NOTEBOOK ALL CLASS C

Date: 25 February 94, 17:26:15 EST
From: STROHMAI at WTSCPOK
To:   VERONICA

Vern,
The train leaves Beacon at half past eight. So let's meet for
breakfast at seven.

Martin
...
```

Figure 57. NOTE with New CLASS Option

3.5.3 DEFAULTS Command

The DEFAULTS command is used to set up default options or display the current default options for several commands. With the DEFAULTS command, you can change the default class used for the NOTE, NETDATA SEND, or the SENDFILE command. Valid classes are A though Z, 0 through 9, or an equal sign (=). If you specify an equal sign (=), the current CP punch class is used when sending a file.

In Figure 58, you are shown all the options for setting the defaults for note. Notice the CLASS option.

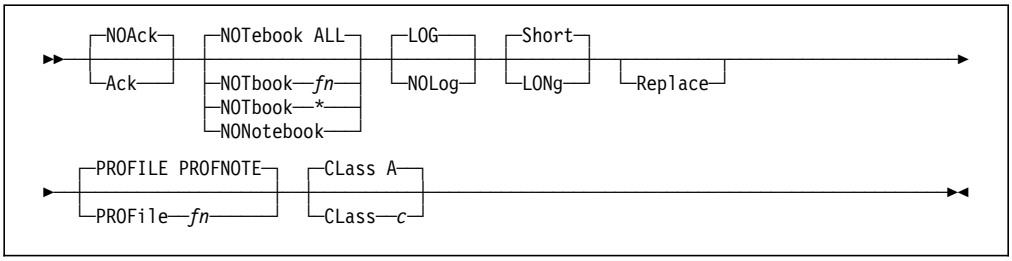


Figure 58. DEFAULTS Options for the Note Command

3.5.4 CMS NETDATA Command

The operand SEND of the NETDATA command is used to send files or notes to a user ID at a network node in Netdata format. This command uses the same values as on the SENDFILE command (A-Z, 0-9, =).

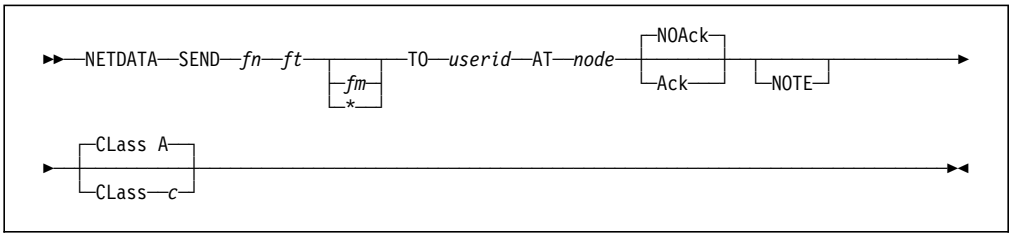


Figure 59. NETDATA Command Syntax with CLASS Option

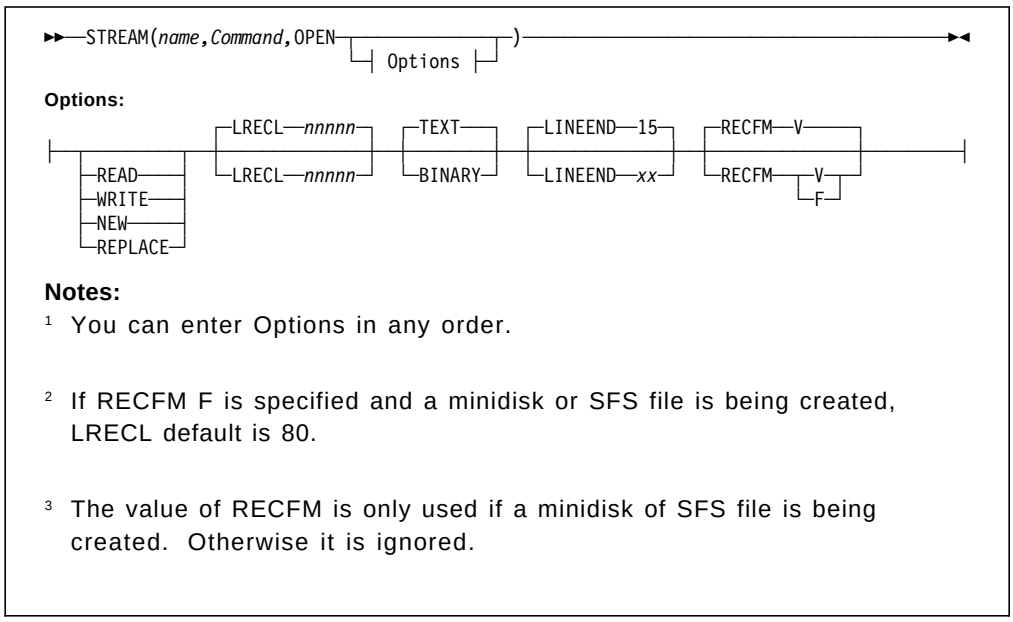
3.6 Support for Fixed RECFM Files in REXX

When REXX Level 2 I/O functions were introduced in VM/ESA Release 2.1, the creation of fixed-record-format SFS and minidisk files was not supported. After the introduction of these functions, it was discovered that many customers require the ability to create new fixed-record-format files for use with program products and applications that are written to use only fixed-format files. For this reason, VM/ESA Release 2.2 adds to the REXX Level 2 I/O functions the support for the creation of fixed-record-format SFS and minidisk files.

3.6.1 REXX STREAM Function Update

The REXX STREAM function, which controls the physical characteristics of a file, was modified to include the RECFM option on the OPEN command. The operands V and F designate variable and fixed record format file options.

Figure 60 shows the updated REXX Stream function.



Notes:

1 You can enter Options in any order.

2 If RECFM F is specified and a minidisk or SFS file is being created, LRECL default is 80.

3 The value of RECFM is only used if a minidisk of SFS file is being created. Otherwise it is ignored.

Figure 60. REXX Fixed Record Format File Syntax

3.6.2 REXX Fixed Record Format Example Program

In Figure 61, the contents of a small REXX program, called FIXED, is shown. This code will use the REXX STREAM function to create a fixed record file with a record length of 90. The output file does not exist before the program is run. The sample program shows how to use a file handle in place of the conventional filename as input to the LINEOUT function. Also, the program shows how REXX I/O routines can be called as both functions and internal subroutines.

After the code sample, a STATE command is used to determine if the output file, MYFN MYFT A, exists. Then, FIXED EXEC is executed. Through the LISTFILE command, you can determine that the file is a fixed-format file, and the record length matches that specified in the program. A CMS TYPE command is used to display the contents of the file. Since a record length of 90 characters is too great to fit on a screen, the line wraps and the extra characters are shown as a blank line in the output.

For more information, please see the *VM/ESA REXX Reference*.

```
type fixed exec

/* Creating a new fixed-format file named MYFN MYFT A  */

/* Create new fixed recfm file MYFN MYFT A using Stream function */
res = Stream('MYFN MYFT A','C','OPEN NEW LRECL 90 RECFM F')
parse var res ok handle .        /* get results from Stream function */

/* Write one record to our new file - the result of the lineout will */
/* be in the variable result                                         */
Call Lineout handle,'REXX is now FIXED'

/* Display file handle and return values from Stream and Lineout */
say x2c(c2x(handle)) ok result
return

Ready; T=0.01/0.01 14:14:38

state myfn myft a

DMSSTT002E File MYFN MYFT A not found
Ready(00028); T=0.01/0.01 14:14:41

fixed

 0000000000 READY: 0
Ready; T=0.01/0.01 14:14:46

listfile myfn myft a (all

FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS
MYFN     MYFT     A1 F         90          1          1
Ready; T=0.01/0.01 14:14:54

type myfn myft a

REXX is now FIXED

Ready; T=0.01/0.01 14:14:59
```

Figure 61. REXX Fixed Record Format Example Program

3.7 Miscellaneous Enhancements to CMS

This section outlines several of the small enhancements pertaining to CMS.

CMS Record Manager Enhancement

VM/ESA Release 2.2 is enhanced to reduce the possibility of a CMS ABEND during record updating of a CMS file.

In previous releases of VM/ESA, when you did a record write to a file and CMS exhausted the virtual storage for that user ID, CMS could not return the file to its original condition. As a result, rather than corrupt the file, CMS would ABEND. CMS now estimates the amount of virtual storage required to perform the write operation.

CMS Synonym Enhancements

The restriction that all synonym files must have a file type of SYNONYM is removed in VM/ESA Release 2.2. The SYNONYM command was updated to reflect this.

PEEK Enhancement

Operation of the PEEK utility for viewing files in the spool has been modified in VM/ESA Release 2.2. PEEK now uses diagnose X′F8′ as the preferred method to find the origin of the file, just as RDRLIST already does. If the sender of the file did not use diagnose X′F8′, the TAG information is read to determine the originating user ID and node. This is a continuation of the Secure File Origin enhancements to CP, CMS, and RSCS.

Console Fullscreen/Linemode I/O Flag

In VM/ESA Release 2.2, the linemode/fullscreen flag is exposed as an Applications Programming Interface (API). This allows a fullscreen application to determine whether or not to reformat the screen due to linemode I/O occuring; for example, CP messages. This reduces the need for applications to refresh the screen unnecessarily.

The linemode/fullscreen I/O flag is available as the CQYDLINE flag in macro CQYSECT. This macro is described in *VM/ESA CMS Application Development Reference for Assembler*.

Pre-allocated Save Areas

In previous releases of VM/ESA, CMS would pre-allocate six save areas that were used for SVC processing. When available, these save areas avoid four calls to CMS storage management. With this enhancement, the number of save areas is increased. This has the potential of improving throughput in a CMS-intensive environment.

3.7.1 Cross Component Enhancements

This section describes enhancements made to other components of VM/ESA Release 2.2 that can be exploited by CMS users. The reader is referred to other sections of this book, where the enhancements are described in more detail.

XEDIT/COPYFILE Usage of R/O Directories

The Shared File System (SFS) has been enhanced to allow the user to decide whether to respect the attribute of R/O directory access for files to which they have explicit write authority.

Please refer to section 5.2, “XEDIT/COPYFILE R/O for Shared File System Files” on page 114, for further information.

SET/QUERY FILESPACE Command

The SET FILESPACE allows the user to change the default file space that is searched when no user ID is defined.

Please refer to section 5.4, “SET and QUERY Filespace” on page 115, for further information.

Diagnostic Tools

Several new diagnostic tools have been provided as samples with VM/ESA Release 2.2. These tools are described in section 7.4, “VM Diagnostic Tools” on page 128.

Addition to Alldates Option of CMS LISTFILE

The ALLDATES option of LISTFILE displays the date and time of creation, the date and time of last update and, with the DTOLC option, the date and time of last change for objects in an SFS directory. Revoked and erased aliases as well as minidisk objects will have a dash (-) appear instead of a date when this option is used. If the server is at a level prior to 2.2, the DTOLC will be displayed as zeros.

The DTOLC, DTOLU, DOLR, and DTOC options determine which fields are in the records produced by the LISTFILE (ALLDATES command. The options are as listed in priority order:

DTOLC Date and time the object was last changed

DTOLU Date and time the object was last updated

DOLR Date the object was last referenced

DTOC Date and time the object was created

The date and time of last change are maintained by SFS to support backup processes, such as DFDMS/VM. Unlike date and time of last update, this data cannot be changed by users, administrators, or applications. For a further discussion of this topic, see section 5.1.3, “Date and Time of Last Change” on page 112 or *VM/ESA CMS Command Reference*.

Enhancement to DMSVALDT CSL Routine

The DMSVALDT Callable Services Library (CSL) Routine provides information about a file that exists either on a CMS minidisk or within a directory owned by the Shared File System.

This routine was enhanced to accept a NAMEDEF. A NAMEDEF is a one to sixteen character variable that can be created using the CMS CREATE NAMEDEF command. The NAMEDEF saves the applications programmer from having to specify for each file to be validated:

- Filename
- Filetype
- Filemode or directory name.

To query any active NAMEDEFs, the user can issue the CMS QUERY NAMEDEF command.

For additional information about the DMSVALDT routine, see *VM/ESA CMS Application Development Reference*.

For additional information about the CMS CREATE NAMEDEF and CMS QUERY NAMEDEF commands, see *VM/ESA CMS Command Reference*.

Chapter 4. VMSES/E Enhancements

VM Serviceability Enhancements Staged/Extended (or VMSES/E for short) is the tool designed to install and maintain VM/ESA. As time progresses, it will also become the standard mechanism for installing and maintaining VM program products. Already, several IBM licensed program products are available that are VMSES/E installed and maintained. These products are detailed in Table 6.

Table 6. IBM Licensed Program Products that are VMSES/E Enabled

Product Number	Product Description	Availability Date
5648-020	Automated Data Storage Management (ADSM) V1.1.0	07/93
5648-038	Software License Monitor (SLM/VM) V1.1.0	03/93
	Software License Monitor (SLM/VM) V1.1.1	06/93
5648-039	Lan File Services/ESA (LFS/ESA) V1.1.0	10/93
5654-007	Automated Networking Operations (ANO) V1.1.0	06/93
5684-096	RSCS V3.1.1	09/93
5684-096	RSCS B1 Security Feature	03/93
5684-100	Pass-Through Facility/VM (PVM) V2.1.1	06/93
5684-120	Workstation LAN File Services (WLFS) V1.1.1	03/93
5684-137	VM/Batch V2.2.0	11/93
5684-142	LAN Resource Extension and Services (LANRES) V1.2.0	03/93
	LAN Resource Extension and Services (LANRES) V1.2.1	06/93
	LAN Resource Extension and Services (LANRES) V1.2.2	12/93
5684-143	Search Manager V1.2.0	2/94
5684-157	Host Management Facility (HMF) V1.1.1	12/93
5684-168	GDDM/VM V3.1.1	6/94
5686-037	VSAM V2.2.0	06/93
5688-188	C/370* Specific Library V2.2.0	06/94
5688-198	AD/Cycle LE/370 V1.3.0	6/94
5688-216	AD/Cycle C/370 Compiler V1.2.0	06/94
5688-226	Engineering Scientific Subroutine Library (ESSL) V2.2.0	1/94
5698-DWD	Real Time Monitor (RTM/ESA) V1.5.2	12/92

With each release of VM, new enhancements are made to VMSES/E. This chapter describes the enhancements made to the operation and functionality of VMSES/E in Release 2.2 of VM/ESA. These enhancements include new function and performance improvements over previous releases of VMSES/E.

The ITSO document on VMSES/E, *ITSO VMSES/E Primer: Concepts and Experiences* (GG24-3851), is highly recommended for anybody who needs a detailed knowledge of the workings of VMSES/E.

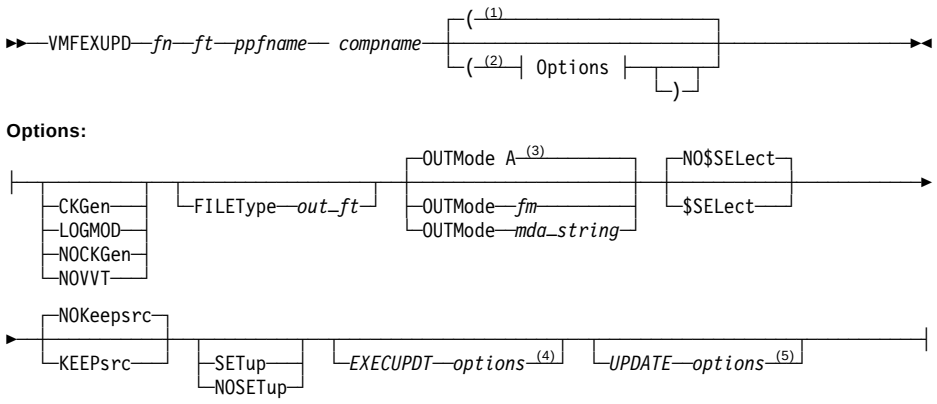
4.1 New VMSES/E EXEC Update Function

VMFEXUPD is a front-end EXEC that calls EXECUPDT to apply updates to a source program (filetype prefixed by $). This simplifies the task of applying local modifications to an installation's EXECs and XEDIT macros.

These modifications can now be tracked by VMSES/E, using its service inventory. The systems programmer can be informed automatically if any service received onto the system will impact the local modifications that are in place.

VMFEXUPD will automatically unpack the source file and apply the local modifications. It then invokes EXECUPD. A log is kept of any messages that may be issued during the application of the local modification. The user can also select whether to update the software inventory with details of the local modification. This option allows for testing of the local modification, and only updating the service inventory once the component has completed testing.

The syntax of the VMFEXUPD command is provided in Figure 62.



Notes:

1. The defaults you receive appear above the line in the options fragment.
2. You can enter options in any order between the parentheses.
3. The output mode for the executable version of the source program. If KEEPSRC is specified, it is the output mode of the merged source.
4. The VMFEXUPD EXEC accepts all EXECUPDT command options and calls the EXECUPDT command with the HISTORY and SID options.
5. The VMFEXUPD EXEC accepts all update command options.

Figure 62. VMSES/E VMFEXUPD Command Syntax

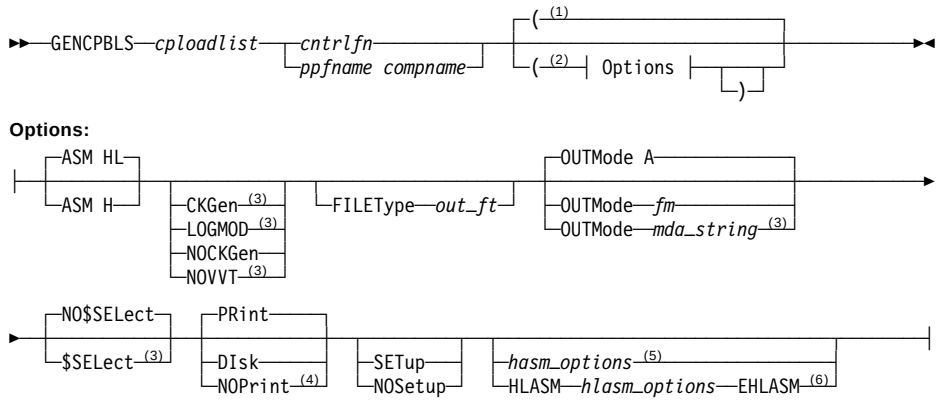
In Appendix B, "Local Modification to SYSPROF EXEC" on page 147, the application of a local modification to the SYSPROF EXEC is illustrated. This example uses the VMFEXUPD command and demonstrates several of the options available on the command.

For additional information about the VMFEXUPD command, please refer to *VM/ESA VMSES/E Introduction and Reference*.

4.2 Automated Update of CP Nucleus Buildlist

In previous releases of VM/ESA, any local modification to the CPLOAD buildlist had to be performed manually. This involved many manual steps and was prone to error. VMSES/E did not have any automatic mechanism for tracking these updates. Should VM service modify the buildlist, a local modification may have been backed out, unknown to the user.

VMSES/E, in Release 2.2 of VM/ESA, provides the GENCPBLS EXEC. This EXEC automates the procedure for applying local modifications to the CP nucleus buildlist. The syntax of the GENCPBLS is shown in Figure 63.



Notes:

1. The defaults you receive appear above the line in the options fragment.
2. You can enter options in any order between the parentheses.
3. The CKGEN, LOGMOD, NOVVT, OUTMOD *mda_string*, and $SELECT options cannot be used with the *cntrlfn* operand.
4. The NOPRINT option is only valid when you also specify the ASM HL option.
5. You can only enter *hasm_options* when you use the ASM H option.
6. You can only enter *hlasm_options* when you use the ASM HL option. When you enter HLASM options, begin with the HLASM keyword and end with the EHLASM keyword.

Figure 63. VMFSES/E GENCPBLS Command Syntax

The procedure for applying a local modification to the CP nucleus build list is described in *Appendix F: Updating the CP Load List* of the *VM/ESA Service Guide*. In this book, we summarize the procedure for modifying the CP nucleus buildlist.

4.2.1 HCPMDLAT Macro

Before making any changes to the CP nucleus buildlist, the user must understand the part HCPMDLAT plays in the scenario. The HCPMDLAT macro is the CP MoDuLe ATtributes macro. Within this macro are listed all the text decks that will be part of the CP nucleus. This list is arranged in a very specific order, as it details the order in which text decks will be loaded at IPL time. The order of sections within the macro are as follows:

1. Resident (that is non pageable) modules
2. Modules used for initialization, but that can be paged out following this operation
3. Pageable modules that are not required during initialization, but can be paged in and out of real storage as required
4. Modules capable of running in a multiprocessor environment
5. Modules not capable of running in a multiprocessor environment.

If you need to update the CP nucleus buildlist, you will need to put an entry in the relevant section of the HCPMDLAT macro. When making modifications to HCPMDLAT, the base macro file should not be modified. The AUX file structure should be used to track the updates to the source file.

Once you have created an update to the HCPMDLAT macro and identified it in a HCPMDLAT AUXLCL, you simply invoke the GENCPBLS EXEC to update the CP nucleus buildlist. This EXEC will perform the following tasks:

1. Update the local Version Vector table to signify a local modification has occurred to HCPMDLAT macro and CPLOAD EXEC
2. Create a temporary MACLIB on the A-Disk containing the updated HCPMDLAT macro
3. Assemble HCPLDL (the CP Loader List) using the new HCPMDLAT macro as input
4. Create the new CP nucleus buildlist (CPLOAD EXEC) from the output of the HCPLDL assembly
5. Update the CP $SELECT file to note the CP nucleus buildlist and the HCPMDLAT macro have changed.

Important Note

The GENCPBLS EXEC does not rebuild the HCPOM1 MACLIB nor the CP nucleus. Instead, it indicates the parts that need to be rebuilt. The MACLIB and the CP nucleus will only be re-built when the VMFBLD command is issued with the correct options.

VMSES/E now knows that a local modification has taken place to the HCPMDLAT macro and the CP nucleus buildlist and that they need to be rebuilt. If any future VM service needs to be applied to them, VMSES/E will signal that the local modification may need to be reworked.

VM/ESA VMSES/E Introduction and Reference has detailed information about the command syntax of the GENCPBLS EXEC.

4.3 VMSES/E Segment Map Function Enhancements

The VMFSGMAP VMSES/E command was introduced in VM/ESA Release 2.0. This command allows you to:

- Display saved systems, discontiguous saved segments, segment spaces and segment members
- Display virtual storage gaps and overlays
- Differentiate planned versus already built saved segments
- Define or delete saved segments
- Change the characteristics of a saved segment.

Several enhancements have been made to the VMFSGMAP command in VM/ESA Release 2.2. These are detailed in the following sections. The syntax for invoking the command has not changed. All enhancements documented here are available once VMFSGMAP has been executed.

4.3.1 VIEW Subcommand

The VIEW subcommand has been introduced in this release of VMSES/E. This command can be specified with one of the following options:

ALL Displays information for all the segments defined on the system.

SEGDATA Displays information about the segments defined in the SEGDATA file. Segments that were defined and saved without using VMSES/E are not displayed.

ERROR Displays information about all segments the have a class of E.

A sample of the VMFSGMAP screen displaying information about all the segments is shown in Figure 64.

```
                            VMFSGMAP - Segment Map                  More: +
                                                                Lines 1 to 18 of 93

Meg             000-MB          001-MB          002-MB          003-MB
St Name    Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M CMS      SYS W-W-------------1.............2.............3.............
M GCS      SYS W---------------1.............2.............3.............
M PSEG1    DCS 0.............1.............2.............WWWWWWWWWWWWWWWW
M MONDCSS  DCS 0.............1.............2.............CCCCCCCCCCCCCCC>

Meg             004-MB          005-MB          006-MB          007-MB
St Name    Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M SNASEG   SPA 4.............================================7.............
M VTAM     MEM 4.............RRRRRRRRRRRRRRRRRRRRRRRR.......7.............
M NPM      MEM 4.............5.............6........RRRR...7.............
M NETVSGOO MEM 4.............5.............6............RRR7.............
  CMSPIPES DCS 4.............5.............6.............RRRRRRRR--------
M GCS      SYS 4.............5.............6.............RRRRRRNNNNNNNNN>
M MONDCSS  DCS >CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

Meg             008-MB          009-MB          00A-MB          00B-MB
 F1=Help     F2=Chk Obj   F3=Exit      F4=Chg Obj   F5=File       F6=Save
 F7=Bkwd     F8=Fwd       F9=Retrieve  F10=Add Obj  F11=Del Obj   F12=Class
====>
```

Figure 64. Display of VMFSGMAP Segment Map Screen

4.3.2 Display Status Code of Segments

The PF12 key has been enhanced so that it toggles between displaying the status of the segments or the class of the segments. This information is displayed in column 1 of the VMFSGMAP screen. Possible spool status codes are:

D Different: the definition on the system does not match the definition in the SEGDATA file.

E Error: the system definition, or the SEGDATA definition is in error.

M Mapped: the saved segment or saved system is defined on the system, but does not appear in the SEGDATA file.

P Planned: the saved segment or saved system is defined in the SEGDATA file, but is not on the system.

blank The definition on the system matches the definition in the SEGDATA file.

Figure 64 on page 95 shows the VMFSGMAP screen in STATUS mode.

Pressing PF12 will toggle the screen to spool class mode. Possible spool class codes, which mimic standard CP NSS class codes, are:

A Active: the saved segment or saved system has been defined and code has been loaded into the saved segment or system.

S Skeleton: a skeleton has been defined, but no code has been loaded into the saved segment or system.

R Restricted: code has been loaded into this saved segment or system. However, this code is only available to users with the NAMESAVE option in their directory.

blank Saved segment does not exist on the system.

4.3.3 Check Object Enhancement

Check object (PF2 CHK OBJ on the VMFSGMAP screen) has been enhanced to provide the display of the CP QUERY NSS command output. The highlighted areas in Figure 65 show the changes to this interface.

```
                          VMFSGMAP - Segment Map                 More: +
                                                              Lines 1 to 18 of 93
 ------------------------------------------------------------------------------
                         Query NSS Map For CMSPIPES
                                                                Lines 1 to 3 of 3
 FILE FILENAME FILETYPE MINSIZE  BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
 0129 CMSPIPES DCSS       N/A    00700  0077F   SR  P  00025   N/A       N/A
 0167 CMSPIPES DCSS       N/A    00700  0077F   SR  A  00002   N/A       N/A

 F1=Help F3=Exit F6=File Query F7=Bkwd F8=Fwd F9=Retrieve F12=Cancel
 ------------------------------------------------------------------------------

Meg             000-MB          001-MB          002-MB          003-MB
St Name    Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789
M CMS      SYS W-W-------------1.............2.............3.........
M GCS      SYS W---------------1.............2.............3.........
M PSEG1    DCS 0.............1.............2.............WWWWWWWWWW
M MONDCSS  DCS 0.............1.............2.............CCCCCCCCCC
Meg             004-MB          005-MB          006-MB          007-MB
VMFSMD2032I System and SEGDATA definitions are the same for segment CMSPIPES
 F1=Help     F2=Chk Obj   F3=Exit      F4=Chg Obj   F5=File       F6=Save
 F7=Bkwd     F8=Fwd       F9=Retrieve  F10=Add Obj  F11=Del Obj   F12=Class
====>
```

Figure 65. Display of VMFSGMAP Check Object Screen

4.3.4 Enhancements to Delete Function of VMFSGMAP

In previous releases of VMSES/E, when delete object was selected from the VMFSGMAP panel (using PF11 DEL OBJ key), the range of the segment was replaced by “deleted” in the SEGDATA file. This enhancement marks the segment as deleted within the SEGDATA file. However, it retains the range so that if the segment needs to be reinstated, VMSES/E already has the necessary information to redefine the segment.

Since all additions and deletions in other areas of VMSES/E are handled by the VMFBLD function, for better consistency the actual deletion of the marked segments is now handled in the same way.

4.3.5 Improved Exit from VMFSGMAP

VMFSGMAP was enhanced to remove the possibility that data could be lost inadvertently on exit from VMFSGMAP. If any object is modified, deleted, or added and the data was not saved, the following message will be produced on exit:

```
VMFSGM2035R Segment data has been changed. Enter (1) if you want
            to exit without saving the changes or (0) to return.
```

4.4 Segment Build Enhancements

The VMSES/E segment build part handler (VMFBDSEG) was modified to use some of the enhancements introduced in VM/ESA Release 2.2. Some of these enhancements may have been documented in other sections of this manual. Where this is the case, the reader will be referred to the relevant section.

Enhanced SYSTEM SEGID Support

In this release, CMS SYSTEM SEGID support was enhanced. See section 3.3, “SYSTEM SEGID Enhancement” on page 80 for CMS information on this topic.

VMFBDSEG will exploit this enhancement. After rebuilding any PSEGs, VMFBDSEG will issue VMFBDS2003W (move SYSTEM SEGID file to the S-disk) only if it determines that the changes to the SYSTEM SEGID file require it.

Enhancement to Segment Space Definition and Build

The segment space building function of VMFBDSEG has also been enhanced in this release of VM/ESA. When rebuilding a member of a segment space, the range of unchanged members of the space are replaced with the range of the member being built. This modification was made to detect the possibility of a build failure when a new segment overlaps a segment skeleton within the segment space.

Rebuild Licensed Products Segments

With previous releases of VMSES/E, it has proven difficult to build segments for different products from the one command entry. The reason for this has been that each product requires access to its own disks, and often products use the same virtual address.

VMFBDSEG was modified to add new options that will allow the linking and detaching of product minidisks, as that product's segments need to be built. Details of these options are:

ACCESS
Invokes VMFSETUP to establish the access order when building a segment. The disks are assumed to be already linked. The disks will be released once the build operation has completed. This was the default option for all previous releases of VMSES/E.

LINK
Links to the disks specified in the product parameter file (PPF). All tasks detailed in the ACCESS option above are then performed. The disks are detached from the virtual machine once the build operation has completed.

NOACCESS
This option assumes that the disks are already linked and accessed, it will simply perform the build operation.

4.5 Build Function Enhancements

With each release of VM/ESA, more components of the operating system are built automatically using VMSES/E's VMFBLD command. Also, some licensed program products are now installed and serviced using VMSES/E. As a result, the VMFBLD command has been enhanced to provide better performance. Also, new options allow the system programmer to test new components or objects with greater flexibility.

4.5.1 VMFBLD Performance Enhancement

A component can be rebuilt with one of the following options:

STATUS
Identify any components or objects that have been installed or serviced and need rebuilding.

SERVICED
Perform all actions of the STATUS option, and rebuild any components or objects that have been serviced.

ALL
Perform all actions of the STATUS option, and rebuild any components or objects specified on the command line, whether they have been serviced or not.

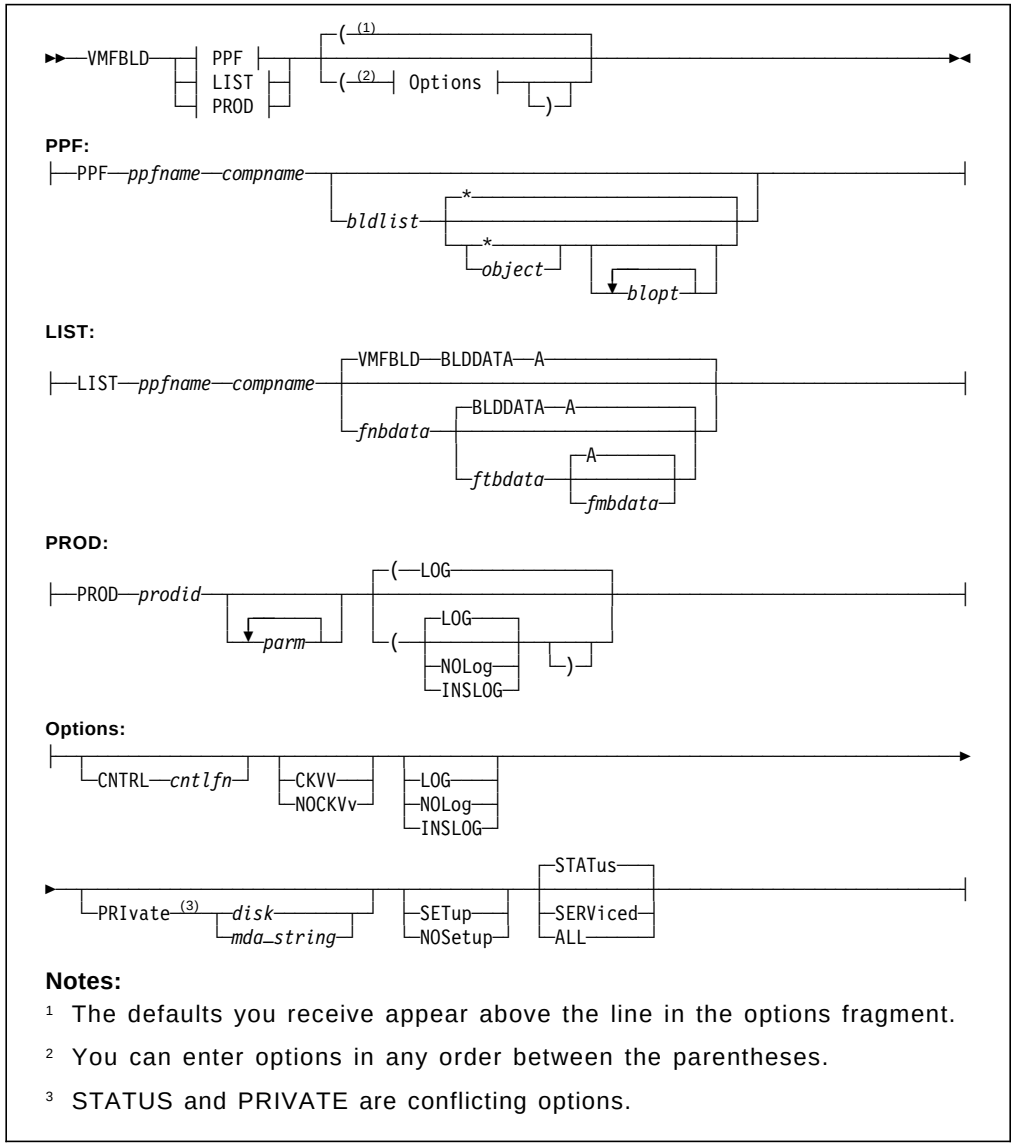
As can be seen, the STATUS option is a pre-requisite to all the other build options. With VM/ESA Release 2.2, any redundant checking that was performed when the SERVICED or ALL options were used has been removed.

Also, if no data appears in the $SELECT file, the status step will not be executed. The $SELECT file is the file that is updated to flag that a change has occurred to a file within that component. This change may have been introduced as part of a VMSES/E service application.

This enhancement allows VMFBLD to perform more efficiently.

4.5.2 New Options on VMFBLD Command

Figure 66 shows the syntax of the VMFBLD command. The options PRIVATE and LIST are new options added with VM/ESA Release 2.2.



Notes:

1. The defaults you receive appear above the line in the options fragment.
2. You can enter options in any order between the parentheses.
3. STATUS and PRIVATE are conflicting options.

Figure 66. VMSES/E VMFBLD Command Syntax

PRIVATE Option of VMFBLD Command

This option allows the system programmer to build an object onto the disk specified. The VMSES/E service inventory files **will not** be updated to reflect this build.

The PRIVATE option can be used to test build an object, and when all the testing completes successfully, the VMFBLD command can be reissued without the PRIVATE option so that the service inventory tables reflect the state of the object.

LIST Option of the VMFBLD Command

The LIST option allows the system programmer to build many objects using one VMFBLD command. The objects can be part of one buildlist or different buildlists. By default, file VMFBLD BLDDATA will be searched to get the list of objects to be rebuilt. However, this file can be overridden to suit installation requirements.

The user should be aware that when specifying objects to be rebuilt from different buildlists, all the objects must belong to one component.

4.6 Enhanced RSU Installation Automation

VMSES/E on VM/ESA Release 2.2 was enhanced to ease installation of a Recommended Service Upgrade (RSU) package, by reducing the number of manual steps involved.

There are two methods that can be used to install a RSU tape:

- Load the pre-applied service.
  This method loads the pre-applied service to the disks specified in the delta string of the PPF. Pre-built objects are also loaded to the relevant build disk. With this method, VMFAPPLY and VMFBLD do not need to be run. An exception is when segments or nuclei require building as a result of the service.
- Load the corrective (COR) service.
  This method loads the service to the delta disk. However, the normal VMFAPPLY command must be run to update the service inventory tables. Also, the VMFBLD command must be run to rebuild objects that have been serviced.

Figure 67 shows the contents of an RSU tape.

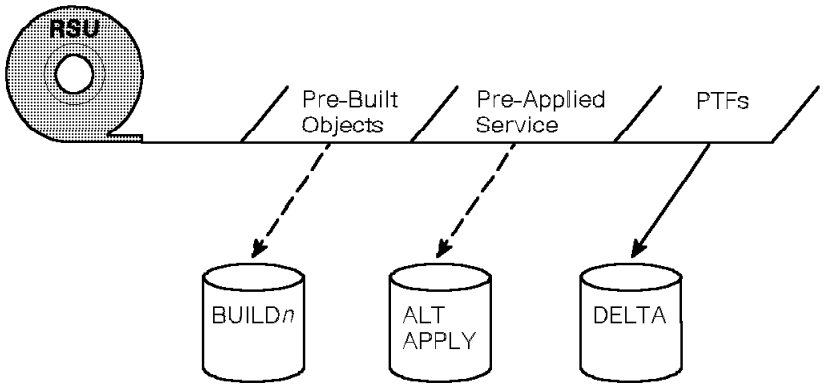


Figure 67. Contents of RSU Tape

In previous releases of VM/ESA, the user had to issue two VMFSIM commands to determine:

- The amount of new service that will be received from the RSU tape.
- The amount of service that must be reapplied to the system after application of the RSU package.

With this information, the user can determine which method to use to apply the RSU to his system. In VM/ESA Release 2.2 a new EXEC, called VMFPSU, is supplied to remove some of the manual steps and help the user make the decision as to which method to use.

VMFPSU's command syntax is shown in Figure 68.

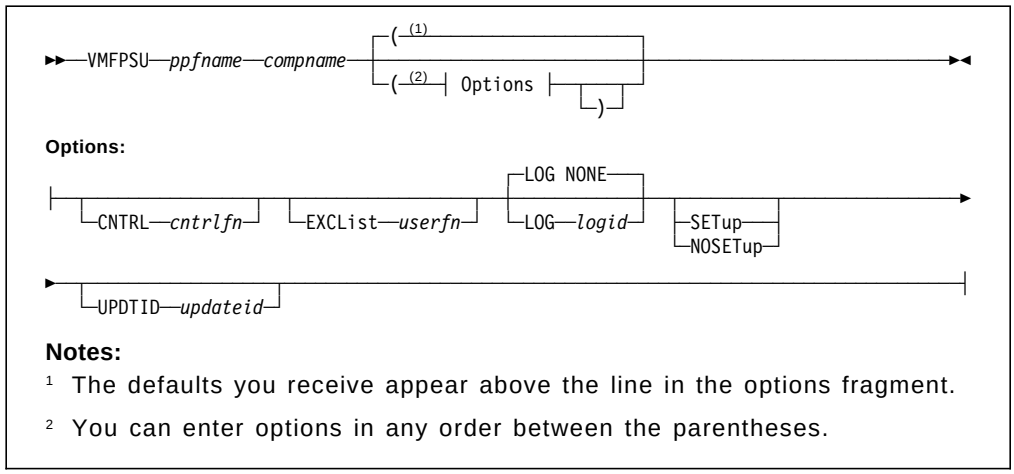


Figure 68. VMSES/E VMFPSU Command Syntax

The VMFPSU EXEC produces a file, called *appid* PSUPLAN, on your A disk. The *appid* is the apply identifier for the component (for example, 6VMVMB22 for the CP component). The *appid* for your component will be given in the *VM/ESA Service Guide*.

The *appid* PSUPLAN contains the following information:

- VMFPSU issued messages
- The number of PTFs on the RSU that are already applied to the component
- List of the PTFs on the RSU that are already applied to the component
- The number of PTFs on the RSU that are not already applied to the component
- List of PTFs on the RSU that are not already applied to the component
- The number of parts that will receive service from the RSU that have local modifications that will need to be reprocessed
- List of the parts that will receive service from the RSU that have local modifications that will need to be reprocessed
- The number of PTFs excluded from the apply list.

Figure 69 shows a sample 6VMVMA22 PSUPLAN file for CMS, created while installing RSU 9404.

```
********************************************************************************
****          PPFNAME: ESA              COMPNAME: CMS                 ****
********************************************************************************
****  PRODID: 6VMVMA22%CMS              Service Level:  201-9404    ****
********************************************************************************
****               Date: 03/18/94  Time: 17:28:26                   ****
********************************************************************************
VMFPSU1071I There are 6 PTFs on the Recommended Service Upgrade for
            PRODID 6VMVMA22%CMS that are not currently
            applied.
VMFPSU1072I There are 0 PTFs currently applied to PRODID
            6VMVMA22%CMS that need to be reapplied.
VMFPSU1076I There are 0 PTFs to be excluded from the Recommended Service
            Upgrade.
VMFPSU1073I There are 1 parts with local modifications that need to
            be reprocessed.
********************************************************************************
****   PTFS TO BE APPLIED FOR PRODID 6VMVMA22%CMS                   ****
********************************************************************************
    PTF.APAR         PTF.APAR         PTF.APAR         PTF.APAR
UMRSU09.VMRSU09  UM06003.VM00573  UM06003.VM00574  UM06302.VM00575
UM06302.VM00576  UM06604          UM06604.VM00578  UM06604.VM00579
UM06604.VM00580  UM06604.VM00581  UM06604.VM00582  UM06702.VM00583
UM07002          UM07002.VM00432  UM07002.VM00442  UM07002.VM00449
UM07002.VM00584  UM07002.VM00585  UM07002.VM00586  UM07002.VM00587
********************************************************************************
****   PTFS TO BE REAPPLIED TO PRODID 6VMVMA22%CMS                  ****
********************************************************************************
NONE
********************************************************************************
****   PTFS EXCLUDED FOR PRODID 6VMVMA22%CMS                        ****
********************************************************************************
NONE
********************************************************************************
****   LOCALMODS TO REPROCESS: 6VMVMA22 VVTLCL    ON DISK 3C4(E)    ****
********************************************************************************
PART - SYSPROF EXC
  PTF - UM06302.VM00576
  MOD - LC00001.LCL00001
```

Figure 69. Sample 6VMVMA22 PSUPLAN File

Notes:

1. The user must issue VMFINS INSTALL INFO prior to executing the VMFPSU EXEC. This ensures that all the Version Vector tables are available when VMFPSU is run.

For more information on these options, refer to *VM/ESA VMSES/E Introduction and Reference*.

4.7 PPF Compile Enhancement

The VMFPPF EXEC updates and compiles a source product parameter file into its usable form. In this release of VM/ESA, VMFPPF has been enhanced so that multiple compiles can be performed with the one command. Figure 70 shows the command syntax for the VMFPPF command.

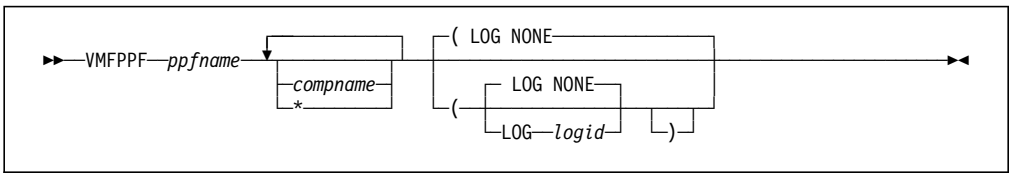


Figure 70. VMSES/E VMFPPF Command Syntax

You can now run VMFPPF with only a *ppfname* as the single option. VMFPPF will search the PPF and list all the components contained within it. The user then has the option of rebuilding some, all, or none of the components. A sample of this is shown in Figure 71.

```
vmfppf esa
VMFPPF2602R The following components can be compiled for ESA $PPF.
            Enter the numbers of your choices separated by blanks
(1) AVS
(2) AVSSFS
(3) AVS370
(4) AVS370SFS
(5) CMS
(6) CP
(7) DV
(8) GCS
(9) GCSSFS
(10) REXX
(11) TSAF
(12) TSAFSFS
(13) VMSES
(14) All the components
(15) Exit
15
VMFPPF2760I VMFPPF processing completed unsuccessfully
Ready(00500); T=0.05/0.07 13:26:08
```

Figure 71. VMSES/E Component Names

Specifying an asterisk (*****) as the second option of the VMFPPF command causes all components within that PPF file to be rebuilt.

In previous releases of VMSES/E, message logging for the VMFPPF command was controlled by the :LOG tag in the :CNTRLOP section of the PPF file. This option specified whether message logging was turned on or off for the command execution. In VM/ESA Release 2.2, message logging is controlled by the LOG option of the VMFPPF command.

For more information on the VMFPPF command, refer to *VM/ESA VMSES/E Introduction and Reference*.

4.8 VMFASM, VMFHASM, and VMFHLASM Enhancements

The assembler routines are enhanced with new options so programmers can better deal with source updates. When making modifications to an assembler routine, the base assembler file should not be modified. The AUX file structure should be used to track the updates to the source file.

The new options provided by these enhancements allow greater flexibility in testing and tracking modifications to assembler code. The command syntax of the three assembler commands varies. You should consult *VM/ESA VMSES/E Introduction and Reference* for this information. However, the new options are common to the VMFASM, VMFHASM, and VMFHLASM commands.

These options are explained as follows:

NOVVT
This option was added to create a test copy of the assembler routine on the user's A-disk. The AUX file structure is used to generate the text deck. No comparison is done with the service inventory tables maintained by VMSES/E. The service inventory tables are not updated to reflect the assembly of this routine.

OUTMODE
This option allows the user to specify onto which accessed disk or directory the text deck should be placed.

$SELECT
VMSES/E tracks when service has been applied to an object or when an object has been rebuilt by updating the $SELECT table for the component that owns the object. This $SELECT table is used by VMFBLD to rebuild any nucleus, library or module that may contain the serviced part.

This option updates the $SELECT for the component to reflect any text decks that are regenerated.

NO$SELECT
This option allows you to assemble a file without causing updates to any objects that use this file.

Please Note

VMFBLD uses the Version Vector tables (VVT) to determine the service level of the part that will be used during build processing. The VVT must be updated to reflect the correct service level. If you need to update the VVT, use the LOGMOD option of the assembly routines.

For more information on these commands, refer to *VM/ESA VMSES/E Introduction and Reference*.

4.9 VMFNLS EXEC Enhancements

The VMFNLS EXEC provided with VMSES/E applies updates to the source National Language Files provided with VM/ESA Release 2.2. VMFNLS was also enhanced to add the same options as VMFHASM.

These options have already been described in section 4.8, "VMFASM, VMFHASM, and VMFHLASM Enhancements" on page 103.

For further details on this command, refer to *VM/ESA VMSES/E Introduction and Reference*.

4.10 VMFINS Enhancements

VMSES/E was designed to aid the user in automating the installation and maintenance tasks on a VM/ESA system. User exits have been provided so that specific installations can modify the default setup to suit their system configuration.

4.10.1 VMFINS Prompt Overrides

One of the possibilities provided is the ability to generate changes or overrides to the IBM supplied product parameter files (PPFs). As licensed products move to being installed by using VMSES/E (see Table 6 on page 91), users may want to override the details provided in the default PPF. In previous releases of VM/ESA, when you issued the VMFINS command for a product, you would receive the following message:

```
VMFINS2601R Do you want to create an override for CP
Enter 0 (No), 1 (Yes) or 2 (Exit)
```

This gave you the option of overriding the defaults. Certain installations will always use the default parameters, and as a result, this message is redundant for them. In VM/ESA Release 2.2, VMFINS has been enhanced to provide new options to handle this prompt. These enhancements have been made to the VMFINS INSTALL option (for installing a new product or completely replacing an existing product) and the VMFINS MIGRATE option (to put a new product on the system while keeping any user tailored files and SFS file authorizations and aliases).

For the complete syntax of these commands, refer to *VM/ESA VMSES/E Introduction and Reference*. In this section, we cover the new options that have been added. These options are common to both commands.

Figure 72 shows the syntax of the new options introduced with this enhancement.

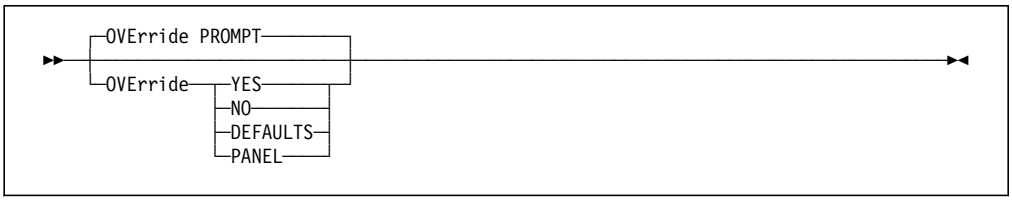


Figure 72. VMFINS Enhanced INSTALL/MIGRATE Options

The OVErride option allows you to control the prompts issued by the VMFINS command. Command line options take precedence over VMFINS DEFAULTS settings. Options for OVErride are:

PROMPT
Asks if you want to create an override for the product you are installing or migrating. If you answer yes to the prompt, which was shown earlier in this section, you are presented with the following message:

```
VMFNKO2917R Do you want to use the defaults for this product?
Enter 0 (No), 1 (Yes) or 2 (Exit)
```

YES
Does not ask you if you want to create an override. You are, however, asked if you want to use the defaults for this product.

NO
Suppresses both prompts and does not display the Make Override Panel.

DEFAULTS
Suppresses both prompts and creates an override using the defaults for the product.

PANEL
Suppresses both prompts and displays the Make Override Panel so you can enter new values for the installation parameters.

For more information on these options, refer to *VM/ESA VMSES/E Introduction and Reference*.

The defaults for VMFINS are stored in the VMFINS DEFAULT file, a sample of which is shown in Figure 73.

```
*************************************************************************
*   COPYRIGHT -                                                 @VR4CRJT
*                                                               @VR4CRJT
*     THIS MODULE IS "RESTRICTED MATERIALS OF IBM"              @VR4CRJT
*     5684-112 (C) COPYRIGHT IBM CORP. - 1990, 1994             @VR4CRJT
*     LICENSED MATERIAL - PROPERTY OF IBM                       @VR4CRJT
*     SEE COPYRIGHT INSTRUCTIONS, G120-2083                     @VR4CRJT
*     ALL RIGHTS RESERVED.                                      @VR4CRJT
*                                                               @VR4CRJT
*   STATUS - VM/ESA Version 1, Release 2.2                      @VR4CRJT
*************************************************************************
*                        VMFINS DEFAULTS                                *
*************************************************************************
ADD                           * default is ADD
NOPLAN                        * default is NOPLAN
MEMO                          * default is MEMO
NORESOURCE                    * default is NORESOURCE
LINK                          * default is LINK
SIDISK       51D              * default is 51D
SIMODE       D                * default is D
DFNAME       USER             * default directory name
DFTYPE       DIRECT           * default directory type
DFMODE       *                * default directory mode
SYSTEM       VM               * default system id is VM
FILEPOOL     VMSYS:           * default is VMSYS:
OVERRIDE     PROMPT           * default is PROMPT
```

Figure 73. Default VMSES/E VMFINS DEFAULTS File

Figure 74 shows a Make Override Panel that was issued as a response to the following command:

vmfins install ppf 5686037a vsaminssfs (override panel resource

This command is used to install the product VSE/VSAM (5686-037) into the Shared File System.

```
   File    Help
_____________________________________________________________________
MKOVR1                        Make Override Panel
                                                                More:
Storage resource for product 5686037A%VSAM component VSAMINSSFS
   Userid................. 5686037A
     VMSES/E WORK DISK....  191         Link as.........  191
     BASE DISK............. VMSYS:5686037A.OBJECT
     SAMPLE/LOCAL FILES... VMSYS:5686037A.LOCAL
     VSAM SERVICE FILES... VMSYS:5686037A.DELTA
     AUX AND INVENTORY FI. VMSYS:5686037A.APPLYALT
     AUX AND INVENTORY FI. VMSYS:5686037A.APPLYPRD
     TEST DISK............. VMSYS:5686037A.BUILD

   Userid................. MAINT
     CMS code required fo.  193         Link as.........  193




Command===>
F1=Help  F2=Command  F3=Exit  F4=Expand Dirid  F5=Save as...  F6=Mdisk or Dirid
F7=Backward  F8=Forward  F9=Retrieve  F10=Action  F11=Conflict  F12=Cancel
```

Figure 74. Make Override Panel for VSE/VSAM Install

4.10.2 Install Function Default File Pool Enhancement

In previous releases of VM/ESA, VMFINS INSTALL used VMSYS as the file pool name on the Make Override Panel when toggling from a minidisk to a Shared File System directory. VMFINS MIGRATE used the Shared File System VMSYS:.*userid*.VMFINS directory for its migration save area. In this case, if resources needed to be set up and privileges granted, it was assumed that VMSYS was owned by user VMSERVS.

In VM/ESA Release 2.2, the VMSYS (and VMSERVS) restriction is removed. A new FILEPOOL option allows you to specify which filepool is used in the above situations. If the owner of the file pool resource, such as VMSYS, needs to be determined, this is done automatically through the QUERY RESOURCE command. Figure 75 shows the syntax of the new options introduced with this enhancement.

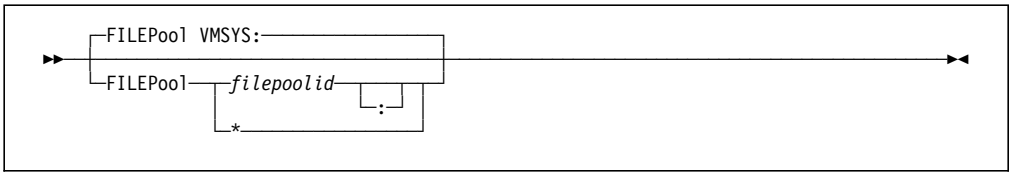


Figure 75. VMFINS INSTALL/MIGRATE Filepool Options

Note:

If you want to permanently override the file pool that will be used by VMFINS INSTALL and VMFINS MIGRATE, you can modify the VMFINS DEFAULTS file. A sample of this file is shown in Figure 73 on page 106.

For more information on these options, refer to *VM/ESA VMSES/E Introduction and Reference*.

4.10.3 VMFINS Usage Notes

Users should be aware the impacts of using the RESOURCE option on the VMFINS command when installing or migrating a product. This option allows VMSES/E to define user IDs and minidisks or directories that will be used during the installation.

Using the resource option requires access to the VMFRMT EXTENTS file (generated during the installation of VM/ESA Release 2.2), and the USER DIRECT file. Users of DIRMAINT or other directory management products should be aware that they will need to create a single file copy of their directory and have it available on an accessed disk before executing the relevant VMFINS command.

It is recommended that if the RESOURCE option is used on the VMFINS command, the directory maintenance virtual machine must be disabled. VMFINS will create updates to the USER DIRECT file and put these online. Should your directory maintenance virtual machine be online simultaneously, there is a possibility that some changes to the directory may be lost.

Once the VMFINS command has completed, the updated directory should be given to your installation's directory maintenance machine.

4.11 Miscellaneous VMSES/E Quality Improvements

This section details miscellaneous quality improvements that have been made to the components of VMSES/E in VM/ESA Release 2.2.

Message Cleanup

Some changes have been made to the VMSES/E message repository, including:

- Removing messages that are no longer used
- Correcting wording on some VMSES/E messages
- Changing help message files to provide clearer information.

VMFOVER Message Logging

The invocation of VMFOVER has been changed so that the user (usually another EXEC) can now specify on the command line whether logging of messages should take place. If it is to take place, the logfile where messages will be written needs to be passed as a parameter to the part handler.

Receive Part Handler for Upper Case Help

As detailed in section 1.8.2, "Installation" on page 12, at installation time you can choose whether upper case help files should be loaded onto the system. A new part-handler (VMFRCUPP) has been introduced to provide this function when service is received.

VMFSETUP Enhancements

The VMFSETUP routine is used by most of the utilities that make up VMSES/E. It is invoked often during installation or service application on a VM/ESA system.

In Release 2.2 of VM/ESA, some enhancements have been made to VMFSETUP. These enhancements allow for easier use of the VMFSETUP Utility.

- VMFSETUP now shows the *ppfname* and the component for which VMSETUP was issued. Please see Figure 76 for an example.

```
vmfsetup esa vmses

VMFSET2760I VMFSETUP processing started for ESA VMSES
VMFUTL2205I Minidisk|Directory Assignments:
            String    Mode  Stat  Vdev  Label/Directory
VMFUTL2205I LOCALMOD  E      R/O  5C4   MNT5C4
VMFUTL2205I LOCALSAM  F      R/O  5C2   MNT5C2
VMFUTL2205I APPLY     G      R/O  5A6   MNT5A6
```

Figure 76. VMSES/E VMFSETUP Command Enhancement

- VMFSETUP has been enhanced in its error detecting routines. Previously, VMFSETUP would only report the first error it encountered on setup (for example an error linking to a disk would be reported, but a subsequent error would not be until the next execution of VMFSETUP).

  VMFSETUP will attempt to link and access all disks and report all the errors it encounters.
- Before setting up the disks, VMFSETUP will check to see if it has a link to the required disk. If the correct disk is linked as the correct address, it will not be re-linked. Similarly, if it is already accessed at the correct mode, it will not be released and re-accessed.

TXTLIB Linkage Enhancements

The VMSES/E part handler that handles rebuilding TXTLIBs (VMFBDTLB) was enhanced in Release 2.2. The enhancements will be used by licensed program products as they become installed and serviced through VMSES/E. The enhancements are as follows:

- Create a TXTLIB from multiple text decks, using the :PARTID option in the buildlist
- ALIAS, NAME, ENTRY, and SETSSI linkage editor statements are now supported on the :OPTION statement in the buildlist.

The *VM/ESA VMSES/E Introduction and Reference* has examples of generating TXTLIB's using these options.

Chapter 5. Shared File System Enhancements

The Shared File System (SFS) was introduced in VM/SP Release 6. It is available as a component of VM/ESA Release 2.2. It allows users to organize their files into hierarchical groups, known as directories. The owner of the files can allow other users access to these files at an individual file level, or at a directory level.

The SFS maximizes the usage of your direct access storage devices (DASD) because, unlike minidisks, user allocated space is used only when needed for information storage. Since the space in the file pool is shared by users enrolled in that file pool, if users do not use their maximum limit of space, the space can be used by other users in the file pool.

This chapter describes the SFS enhancements provided under VM/ESA Release 2.2.

5.1 Overview of Shared File System Backup Enhancements

To provide better and more meaningful information to a backup process, the CMS SFS has changed to do the following:

- Create a file that is marked as migrated. This function assists in restoring a file that was backed up while it was in a migrated state.
- Provide size of authorization data to the backup process. This is used to determine where to place the backup data.
- Provide information relating to the date and time of last change for files and directories.
- Allow direct retrieval of the creation date and creation time of a directory by use of a program interface.
- Provide a method to recover an alias for which a base file does not yet exist.

These enhancements are detailed in the following sections.

5.1.1 Create Migrated File

The QUERY FILEPOOL COUNTER, QUERY FILEPOOL REPORT and QUERY FILEPOOL STATUS commands return a counter for the number of times migrated files were created.

The VMLIB CSL routine DMSOPBLK now provides the ability to create an entry in a SFS catalog that represents a migrated file. A new keyword, CREATEMIG, has been added to DMSOPBLK, which causes a primary repository file to be created or replaced at the commit time of the work unit. CREATEMIG is an administrator function.

DMSOPBLK also returns the current identifier for the primary repository file in a new output parameter, UNIQUE_ID.

5.1.2 Auth Size Output Parameter

The VMLIB CSL routine DMSOPDIR now provides the ability to determine the size of directory authorization information through a new output parameter BUFFSIZE.

5.1.3 Date and Time of Last Change

SFS now provides the date and time of last change for:

- Directories
- Files
- Aliases
- External objects
- File space.

The date and time of last change (DTOLC) is different from the date and time of last update since DTOLC cannot be controlled by a user or an administrator. Updates to DTOLC are performed by the SFS server only. Also, the DTOLC of an object is updated every time there is a change to that object's status, attributes, or contents. DTOLU is what exists on standard minidisk files as well as SFS files. DTOLU can be manipulated with user tools and commands, such as COPYFILE (OLDDATE, to reflect times not consistent with physical changes.

The attributes of DTOLC are:

- Established at commit time.
- Updated when authorizations are granted.
- Top directory is updated when space limits change, that is, when storage is added or deleted.
- Not updated when objects are renamed or relocated within a directory.
- Parent directory is not updated when an object is added or deleted from that directory.
- Not updated for an alias when the base file is updated. Create alias will set the DTOLC for the alias, which will be the date and time that the alias was created.
- Is not seen before commit time. A value of zero is returned if there is no committed value. If a file is closed with NOCOMMIT, then the previous DTOLC is returned.
- Is not updated when commands, such as OPEN and CLOSE, are issued and they do not alter an object. However, if existing data is overwritten with the same data, the DTOLC is updated.
- SFS administration authority is not required to read the DTOLC.
- Is not updated by the file pool restore function.

The DTOLC information may be retrieved by the following:

- VMLIB CSL routines:
  - DMSGETDI (FILEEXT) where DTOLC are new entries in the output buffer
  - DMSGETDX where DTOLC are new output variables
  - DMSEXIST where DTOLC are new entries in the output buffer
  - DMSEXIFI where DTOLC are new output variables
  - DMSEXIDI where DTOLC are new output variables.
- CMS LISTFILE command with the ALLDATES option and the DTOLC keyword.

5.1.4 Retrieve Directory Date and Time of Creation (DTOC)

This information is obtained from the VMLIB CSL routines DMSEXIST and DMSEXIDI.

For DMSEXIST:

Directory data record format of the data buffer returned contains four new fields:

- Creation date in decimal format, dec_cr_date
- Creation time in decimal format, dec_cr_time
- Creation date in character format, cr_date
- Creation time in character format, cr_time.

For DMSEXIDI:

Two output parameters have been added to retrieve date and time information from the data buffer:

- Creation date in character format, create_date
- Creation time in character format, create_time.

5.1.5 Unresolved Alias Support

Every regular alias (not revoked, erased, or unresolved) must have an associated base file. There may be times when a function, such as a file pool restore, would need to reconstruct an alias prior to its associated base file. In such a scenario, the function would want to reserve the ability to establish the link between this unresolved alias and its base at the time when the base is created. An unresolved alias can only be created with the use of a CSL routine. Administrator authority is required to do so. An unresolved alias is created when a create alias CSL routine (DMSCRALI) is issued with the new UNRESOLVED keyword and one of the two following conditions is met:

- The base file exists but the authorization requirements are not met.
- The base file does not exist.

A new option, RESOLVE, is provided by the DMSOPBLK CSL routine. This option indicates that alias resolution is to happen when the file being opened is committed.

At commit time (of the DMSOPBLK), one of the following will occur if an alias is found for the base file:

- If the alias can be resolved, resolution will happen and produce an alias.
- If the alias cannot be resolved, that is, the correct authorizations do not exist, the unresolved alias will be converted to a REVOKED alias. A warning reason code, 61621, is returned to indicate that the alias cannot be resolved.

To remove the unresolved alias, besides creating a new alias to resolve it, the unresolved alias:

- May be erased

  To determine which objects are an unresolved alias, a function can use an EXEC to preform an OPENCAT or READ and erase all the rows with the status of J, or issue FILESERV LIST and search the list for rows with status J, then erase them.
- May have the object catalog row reused.

If an attempt is made to create an object with the same name as an unresolved alias (for example, open a file, copyfile or create a new alias), the unresolved alias′s catalog row is reused to create the new object and the unresolved alias will no longer exist.

For further information, see *VM/ESA CMS Application Development Reference* and *VM/ESA SFS and CRR Planning, Administration and Operation*.

5.2 XEDIT/COPYFILE R/O for Shared File System Files

Note: The following discussion and the new commands pertain to file control directories only.

The Shared File System allows an enrolled user to grant authority to access files and directories owned by that user to other users of the same file pool. These authorizations can be broken down into four categories:

READ Authority
Allows the user to read the contents of a file. READ authority on a directory allows the user to read the directory contents (files and subdirectories), if any, and is required to be able to access a directory.

WRITE Authority
Gives the user all the privileges of read plus the authority to modify, rename, or erase the file. For a directory, write gives the authority to create files in that directory. It does not give authorization to read or write any of the files in that directory.

NEWREAD Authority
Indicates that the user (or users) will automatically receive READ authority for any new base files added to the directory.

NEWWRITE Authority
Indicates that the user (or users) will automatically receive WRITE authority for any new base files added to the directory.

In previous releases of VM, when a user accessed a SFS directory they did not own, the directory was accessed read-only by default, even if the user had write authority. The FORCERW option on the CMS ACCESS command allows you to access the directory in write mode.

However, XEDIT and COPYFILE were modified so that even when the directory was accessed R/O, if the user had write authority, the user could update the file within a directory that appears read only. To specifically address the requirement that XEDIT and COPYFILE respect the read only attributes of the directory, a new CMS SET command has been introduced. Figure 77 shows the syntax of the command.

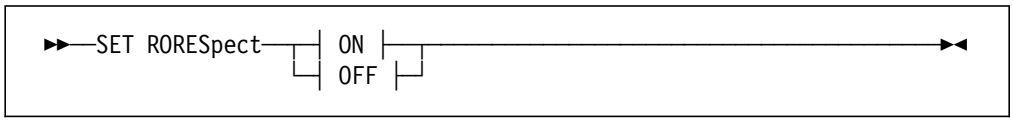


Figure 77. CMS SET RORESPECT Command Syntax

If the setting is ′ON′, any update option to a file in a directory accessed as read only will fail and an error message will be issued.

If the setting is 'OFF', any update option to a file in a directory accessed as read only will be allowed. This maintains compatibility with previous releases of the Shared File System.

The current setting can be queried using the new CMS QUERY RORESPECT command.

For additional information about these commands, see *VM/ESA CMS Command Reference*.

5.3 Shared File System File Pool Rename Enhancement

VM/ESA Release 2.0 introduced the SFS administrator FILEPOOL RENAME command. This command renames the high-level qualifier, or file space, for all the files and directories. This command may be necessary when a VM user ID is changed, but the user still requires access to the files, directories, and authorizations they had under their old user ID.

One of the documented notes for this command was:

> "... A user can gain access to a SFS file pool without being explicitly enrolled if public connect authority exists for the file pool. You can use the FILEPOOL RENAME command to change authorities and aliases for this user by issuing the ENROLL command to enroll the old user ID, then use the FILEPOOL RENAME command to change the old user ID to the new user ID. ..."

The restriction that a user must be enrolled in the file pool before FILEPOOL RENAME will work has been lifted in the Shared File System running on VM/ESA Release 2.2. No change has been made to the command syntax. Testing showed that this function performed correctly.

5.4 SET and QUERY Filespace

The new SET FILESPACE and QUERY FILESPACE commands provided in Release 2.2 of the Shared File System allow all users to change the default file space. This function is similar to the SET FILEPOOL default file pool command provided since the introduction of SFS.

When a user enters the SET FILEPOOL command, either from the CMS command line or through the PROFILE EXEC, the system sets up a default file pool that will be searched when the user issues SFS commands. In Figure 78, we are logged as user ID ORGANK and have issued the SET FILEPOOL command followed by the DIRLIST command. The DIRLIST command is entered without options.

```
set filepool vmsys
Ready; T=0.01/0.01 18:04:49

dirlist
Ready; T=0.01/0.01 18:05:12
```

Figure 78. Shared File System SET FILEPOOL Command

The output of the DIRLIST command is shown in Figure 79.

```
 ORGANK   DIRLIST  A0  V 319  Trunc=319 Size=19 Line=1 Col=1 Alt=0
Cmd   Fm Directory Name
      -  VMSYS:ORGANK
      -  VMSYS:ORGANK.DOCUMENTS
      -  VMSYS:ORGANK.DOCUMENTS.SOURCE
      -  VMSYS:ORGANK.DOCUMENTS.TEXT
      -  VMSYS:ORGANK.DOCUMENTS.REFERENCE
      -  VMSYS:ORGANK.TEST
      -  VMSYS:ORGANK.TEST.DATA
      -  VMSYS:ORGANK.TEST.EXECS


1= Help       2= Refresh  3= Quit   4= Sort(fm)    5= Sort(dir)   6= Auth
7= Backward   8= Forward  9=       10=            11= Filelist   12= Cursor

====>
                                                          X E D I T  1 File
```

Figure 79. Shared File System DIRLIST of Default File Space

Using the SET FILESPACE command we can reset the default user ID to look at another user's directories. If we issue the following command:

```
SET FILESPACE MAINT
```

Assuming we have the authority to access MAINT's directories, a partial output of the DIRLIST command is shown in Figure 80.

```
 ORGANK   DIRLIST  A0  V 319  Trunc=319 Size=19 Line=1 Col=1 Alt=0
Cmd   Fm Directory Name
      -  VMSYS:MAINT.
      -  VMSYS:MAINT.AVS
      -  VMSYS:MAINT.AVS.ALTERNATE
      -  VMSYS:MAINT.AVS.INTERMED
      -  VMSYS:MAINT.AVS.OBJECT
      -  VMSYS:MAINT.AVS.PRODUCTION
                                                          X E D I T  1 File
```

Figure 80. DIRLIST of File Space following SET FILESPACE Command (partial)

The SET FILESPACE command will stay in effect until CMS is re-IPLed within the virtual machine, the user logs off, or another SET FILESPACE command is issued. Directories accessed when the SET FILESPACE command was issued for another user ID will appear in R/O mode, not R/W mode as is the default for your own file space. This is the standard for the access command in the SFS environment.

The format of the SET FILESPACE command is shown in Figure 81.

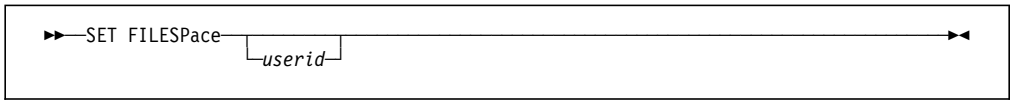


Figure 81. CMS SET FILESPACE Command Syntax for SFS Directories

Where *userid* is the name of the file space that you want as the default file space. If user ID is omitted, the default file space ID is set to the virtual machine ID.

The format of the QUERY FILESPACE command is shown in Figure 82.

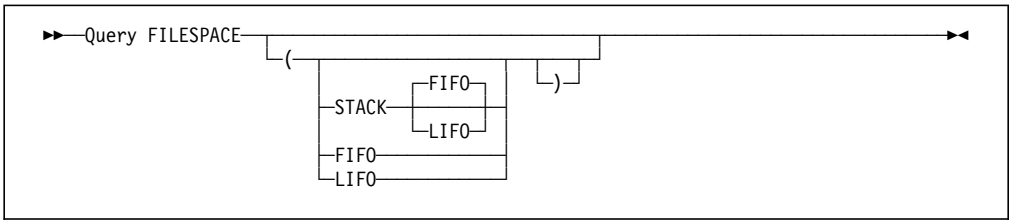


Figure 82. CMS QUERY FILESPACE Command Syntax for SFS Directories

For additional information about these commands, see *VM/ESA CMS Command Reference*.

5.5 Miscellaneous Enhancements to the Shared File System

In the following sections, miscellaneous enhancements to the Shared File System are discussed.

Improved Performance in Revoking Authorities

In VM/ESA Release 2.2, the performance of revoking authorities for files and directories has improved. This will help reduce the amount of time taken by users and the fileserver administrator in performing maintenance and housekeeping functions.

5.6 Miscellaneous Enhancements to Other Components

This section describes enhancements made to other components of VM/ESA Release 2.2 that can be exploited by users of the Shared File System. The reader will be referred to other sections of this book, where the enhancements are described in more detail.

Diagnostic Tools

Several diagnostic tools for the Shared File System have been introduced in this release of VM/ESA. These can help in diagnosing intermittent problems with the Shared File System. Please refer to the sections contained in the following table for details.

Table 7. Shared File System Diagnostic tools

Title	Page	Description
FS2SFSER	129	Report SFS errors that occurred while using FS and OS macros.
LCTRACE	129	Display Linkage Component (LC) requests to SFS server.
PRINTFST	129	Display file status table information.
SFSDOT	130	Documentation for SFS diagnostic aids.

Enhancement to DMSVALDT CSL Routine

The DMSVALDT Callable Services Library (CSL) has been enhanced to allow for NAMEDEFs to be supplied when calling the routine. This routine can be used to validate files on both CMS minidisks and Shared File System directories.

Please refer to section “Enhancement to DMSVALDT CSL Routine” on page 89 for further information on the DMSVALDT CSL.

Chapter 6. Group Control System (GCS) Enhancements

This chapter describes the enhancements made to GCS in VM/ESA Release 2.2. Additional command response information, data space and hardware compression support, and use of Name/Token pairs to reduce GCS outages are some of the enhancements discussed.

6.1 GCS Query Address Enhancements

The GCS QUERY ADDRESS command has been enhanced to provide more information while debugging GCS, which will ease problem identification and resolution.

The following improvements were incorporated:

- QUERY ADDRESS will now give the name of and the displacement into a module or table given a specified address.
- A new QUERY MODDATE command displays the compilation date of a specified module. It can be used to determine if an old level of a text deck has been inadvertently incorporated into the sysgen.
- QUERY MODDATE last specifies that you want a list of all modules compiled on the most recent compilation date.

6.1.1 Query Address Examples

Example 1:

In this example, the two forms of the QUERY ADDRESS command are used with the old *name* option followed by the *address* option.

```
query address gctlos
Address of GCTLOS is 0072F558
Ready;

query address 072f558
Address is GCTLOS + X'00000000'
Ready;
```

Figure 83. GCS Query Address

In this example, the new QUERY MODDATE command is used with the *name* option followed by the *LAST* option. The last option shows the date of the GCTINL text, which is the text used to update the service level for GCS.

```
query moddate gctlos
Date of GCTLOS is 11/04/93
Ready;

query moddate last
Date of GCTINL is 01/28/94
Ready;
```

Figure 84. GCS Query Moddate

For more details about the above mentioned GCS commands, refer to *VM/ESA Group Control System*.

6.2 GCS Level Update

Prior to VM/ESA Release 2.2, only CP and CMS could be queried to indicate their service levels. With VM/ESA Release 2.2, it is also possible to get the GCS service level. Customers knew what release of GCS they were running but had no way to determine what service level of GCS they were currently at. Many customers run with an older GCS service level compared to CP, and often mistakenly report the CP service level to customer support when they are reporting a GCS problem. Or, someone assisting a customer may want a quick way to find out a service level when diagnosing a problem.

To accomplish this, the GCS QUERY GCSLEVEL command and the GCSLEVEL macro have been changed to supply the release information and the service level information.

The GCS service level will be updated in a module that is a table containing the current service level (RSU level). This module is compiled and shipped on each RSU (Recommended Service Upgrade).

In the following example, the enhanced QUERY GCSLEVEL command shows the service level.

```
query gcslevel
VM/ESA Release 2.2, Service Level 401
Ready;
```

Figure 85. GCS QUERY GCSLEVEL

For more details about the preceding commands, refer to *VM/ESA Group Control System* .

6.3 GCS Data Space Support

The GCS data space support has been implemented on GCS to allow future program products the ability to exploit data spaces. Note the use of VM data spaces is only available on hardware that supports access registers.

GCS was enhanced to provide protection and diagnostic tools for applications using access registers. This enhancement includes:

- Preserving the access register through a GCS supervisor call and restoring the registers at the end of the call.
- GCS isolates its internal functions so it will abend whenever a call to the GCS supervisor is made in access register (AR) mode.
- GCS externalizes two flags in the FLS macro. One flag indicates whether the virtual machine is running in XC mode and the other whether the hardware data compression support is supported.
- The SDUMPX macro is provided to allow applications running in AR mode to take a snap dump of a data space. The existing VMDUMP command will continue to be used for dumping your primary address space.

- GCS also supports the latest version of the MVS macro SDWA, which is filled in with the current values of the access registers from the task control block for a task that abends. GCS also supports the MVS mapping macro IHAEPIE used in the ESPIE interface to provide the necessary support to isolate problems in a secondary data space.
- If GCS is IPLed in XC mode, it will set the address space function control bit on in control register 0 allowing applications to issue a SAC 512 to get into access register mode.
- Allowing mixed XA, XC mode virtual machines in a single GCS group. Only XC mode supports the data space environment. (See 6.4, “GCS Name/Token Pair Support” on page 123 for more information).

6.3.1 Additional GCS Data Space and Compression Updates

With the new GCS Data Space support, additional access register checks are introduced.

Synchronous User Exits

Programs returning from the following macros will be followed by a check to make sure the user did not return in access register mode.

GCS Macros	Description
GENIO	Use general input/output devices, except for DASD devices and virtual machine consoles.
IUCVCOM	Use the coordinate communications among users within the IUCV and APPC/VM environment.
IUCVINI	Use to admit or withdraw a user from the IUCV and APPC/VM environment.

New Flags in FLS Macro

GCS provides a few new flags to help an application determine the availability of hardware compression and data space support. These flags are set when a GCS machine IPLs, which allows machines running under a group to be a mix of XA and XC machines.

The FLS mapping macro is updated giving access to the following fields residing in page 0 of virtual storage.

FLSFLG	A fullword that contains two flags to let applications running on GCS know whether the machine is in XC mode and if hardware compression is supported.
FLSFLGXC	This flag will be on if the machine is running in XC mode.
FLSFLGHC	This flag will be on if hardware compression is supported.

6.3.2 SDUMPX

A new SDUMPX macro is available to dump ranges of storage in a data space accessed by an application running on GCS in a XC virtual machine. The SDUMPX macro is available in standard, list and execute formats. Only the standard format is detailed here.

Standard Format

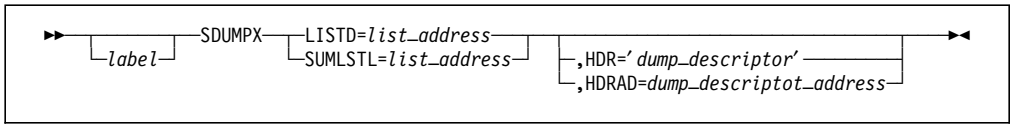


Figure 86. SDUMPX Macro Syntax

Operands

HDR Specifies a string of characters that can be used to describe the dump. The dump description will be displayed by the DVF DUMPSCAN DUMPID subcommand when the LISTD operand of SDUMPX is used to produce the dump.

This character string is placed in the dump to help you identify it quickly. This string can contain up to 100 characters and must be surrounded by single quotation marks.

HDRAD Specifies the address of a string of characters you stored previously that describe the dump. The dump description will be displayed by the DVF DUMPSCAN DUMPID subcommand when the LISTDS operand of SDUMPX is used to produce the dump.

LISTD Specifies the address of a list of address ranges, qualified by an ASIT of 8 bytes, that reference the data space you want to dump. The ASIT uniquely identifies the data space within the scope of the VM system. The ASIT is similar to the STOKEN of MVS/ESA.

SUMLSTL Specifies the address of a list of address ranges, qualified by an ALET of 4 bytes. The ALET is obtained by use of the ALSERV macro which is used to add a data space to a list of data spaces that can be accessed by a given virtual machine. The ALET that is returned from that request can be used to establish addressability to the data space.

Usage Notes

1. When using the SDUMPX macro to dump data spaces, no internal locking is provided.
2. The SDUMPX macro should not be used to dump your primary address space. You should continue to use the SDUMP macro to dump the GCS virtual machine storage where proper locking can be provided to guarantee you get a snapshot of storage without the storage being altered.
3. You must not be in AR mode when you issue the SDUMPX macro. The AR mode form of the SDUMPX macro is not supported on GCS. You must switch off the AR mode before using SDUMPX.
4. It is important to understand the rules governing who receives the spool file containing the dump and what that file contains.

   For security reasons, not every user is authorized to receive dumps containing fetch-protected data. Those who are authorized are listed among the authorized users at GCS build time. If a common dump receiver was specified at GCS build time, then that individual receives the dump. Otherwise, the issuer of the SDUMPX macro receives the dump.

   Bear in mind that if the person receiving the dump is not authorized to handle fetch-protected data, that data will be omitted from the dump.

However, all requested non-fetched-protected data and key 14 storage will be included in the dump.

5. No dump will be produced if dumps are suppressed through the SET DUMP OFF command.

Return Codes and Abend Codes

When this macro completes processing, it passes a return code to the caller in register 15.

Table 8. SDUMPX Return Codes.

Hex Code	Decimal Code	Meaning
X′00′	0	All requested areas have been included in the dump.
X′04′	4	Only a portion of the requested areas was included in the dump.
X′08′	8	A range beyond the last byte of virtual storage was requested, a data space was not properly addressed, the SDUMPX macro was issued from a non-XC virtual machine, or SET DUMP OFF has been issued. (GCS was unable to prevent a dump).

Table 9. SDUMPX Abend Codes.

ABEND Code	Reason Code	Meaning
233	4	Invalid parameter list structure.
233	8	Invalid parameter list address.

6.4 GCS Name/Token Pair Support

Applications running on GCS require that the environment remain up and running for the longest possible period of time.

Since the members of a GCS group are interdependent, termination (especially *abnormal termination*) of a task or virtual machine in a GCS group can leave the group in an undefined state and require taking down the network to correct the problem.

The main purpose of the Name/Token Pair Support allows information to persist past the termination of a task or the reset of a virtual machine, such as the Halt Execution (HX) command, allowing normal operations to be re-established fast and without network outages.

Also, GCS applications can take advantage of this support to improve the efficiency of message passing between virtual machines by bypassing IUCV messaging overhead, thus improving performance.

The GCS Name/Token Pair Support provides a dynamic way to pass pointers or data between:

- Two programs running under the same task
- Two or more tasks running in a single virtual machine

- Different virtual machines in a single GCS group.

It also works for:

- *Private storage:* passing information between two tasks in a single virtual machine. These tokens will persist until they are explicitly deleted, reset, or the virtual machine is IPLed.
- *Common storage:* passing information between two virtual machines. These values persist until the entire group is reset by re-IPLing the recovery machine, and all other machines in the same GCS group.

The API for this function is the new **GCSTOKEN** macro. This macro allows you to create, retrieve, or delete a Name/Token pair. Only authorized applications are allowed to create a Name/Token pair for level=private or common.

For more details about this subject, refer to *VM/ESA Group Control System* .

6.5 GCS Branch Entry Protection

Branch entry protection is implemented to verify whether the applications are disabled for interrupts when branching to GCS code.

If the task is enabled for interrupts when it branches to GCS, the task will abend. A new message will display the information on the branch in error, reducing application debug time, since these errors are difficult to capture.

6.6 Abend X′0F8′ New Reason Codes

Abend X′0F8′ has been enhanced for the new GCS Data Space Support as well as for the Branch Entry Protection Support.

The different reason codes which can appear with abend code X′0F8′, are detailed below.

RC 12 Branch entry protection

If a branch entry to GCS services is called while enabled for interrupts, the GCS supervisor will abend with X′0F8′, reason code 12.

RC 14 Asynchronous user exists

Return from asynchronous exits will be followed by a check to ensure that the user did not return in access register mode.

RC 16 Branch entry fencing

If a branch entry to the GCS supervisor is called in access register (AR) mode, it will result in abend X′0F8′, with reason code 16.

RC 18 SVC call fencing

As GCS fences all calls to the GCS supervisor to ensure that calls are not in AR mode, all the SVC calls to the GCS supervisor in AR mode will result in abend X′0F8′ with reason code 18.

Chapter 7. Problem Determination Aids

While the quality of the code in VM/ESA has increased from release to release, VM/ESA now works in an increasingly complex environment. As you have seen in the earlier chapters of this book, some major enhancements have been made to the features available with VM/ESA.

This chapter details some of the problem determination aids that have been enhanced to help your systems programmer identify the source of any problems that may occur on your VM/ESA Release 2.2 system.

7.1 New SNAPDUMP Function

To create a CP dump, three possibilities are available since VM/SP.

- When CP detects a problem, it takes a dump and restarts itself.
- When you want a dump you can create a restart dump through the RESTART function on the hardware console. This operator action passes a code to CP to create a SVC002 dump and restart itself.
- A Stand Alone Dump (SAD) can be used when a RESTART is impossible, usually as the last resort. This dump is not as easy as a restart dump, since you have to IPL a special disk or tape on which the SAD utility was installed. The dump is not written to the CP spool, but to tape.

The drawback with RESTART dumps and SADs, is that the system needs to be re-IPLed or restarted after a dump has been taken. VM/ESA Release 2.2 introduces the new SNAPDUMP command that allows a dump of system storage to be taken without the need to re-IPL the system on completion. This further reduces the amount of down-time for your system due to problem determination.

SNAPDUMP uses the settings of the CP SET DUMP command to determine which type of dump is taken. It is a class A privilege command. New options have been added to the CP QUERY DUMP command to support the SNAPDUMP function.

Once SNAPDUMP is invoked, it will quiesce all activity on the VM/ESA system while it takes the dump. As a result, the recommendation is that:

- All users be warned before a SNAPDUMP is taken, as they will not be able to use the system until the dump completes.
- A SNAPDUMP be taken outside prime-shift hours.

When taking a full system dump of a system with a large amount of main storage, care should be taken; since the system will be quiesced, communication lines may timeout before the dump has completed. Special operation instructions should be created for this eventuality.

A sample invocation of SNAPDUMP is shown in Figure 87 on page 126.

```
q dump
DASD 0E60 dump unit CP IPL pages 2126
Ready; T=0.01/0.01 09:12:14

set dump e60 all
DASD 0E60 dump unit ALL IPL pages 16416
Ready; T=0.01/0.01 09:12:21

snapdump
HCPSNP9265I The SNAPDUMP command has been entered, causing the non-destructive
dump function to be called.
HCPSNP9265I The system will be quiesced until the dump is complete.
Ready; T=0.01/249.69 09:16:46
```

Figure 87. Sample Use of CP SNAPDUMP Command

When a SNAPDUMP is processed by the DUMPLOAD command, the ABEND code that is assigned to the dump is SNP001.

For more information on the SNAPDUMP command, refer to the *VM/ESA CP Command and Utility Reference* . For information on the DUMPLOAD command and the procedure for processing dumps from your VM/ESA system, refer to *VM/ESA Dump Viewing Facility*.

7.2 CP SET ABEND/QUERY ABEND Enhancements

With the introduction of SNAPDUMP capability, new options have been added to the CP SET ABEND command. The SET ABEND command is used to force soft errors to cause a full system termination or a full system SNAPDUMP.

The command syntax is shown in Figure 88.

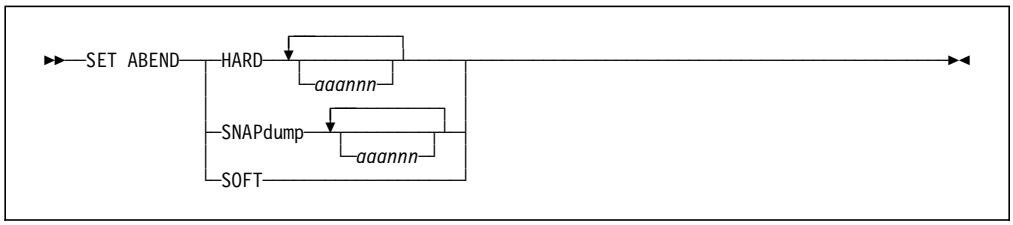


Figure 88. CP SET ABEND Command Syntax

where **aaannn** is the CP ABEND code.

Soft ABENDS are ABENDs taken when an error condition can be isolated to a single virtual machine or when system integrity is not endangered.

The output from QUERY ABEND has also changed to support SNAPDUMP. An example of SET ABEND and QUERY ABEND usage is shown in Figure 89. Here, we are going to change CP ABEND handling so that if soft ABEND AAS001 occurs, a SNAPDUMP of the whole system should be taken. During the dump, the system will be quiesced. Once the dump is complete, normal system operation will restart. The dump will be in the reader of the virtual machine that

receives dumps. This user ID is specified in the SYSTEM CONFIG file or HCPSYS.

If the dump user ID specified in the SYSTEM CONFIG file or HCPSYS is not specified in the CP directory, then all soft abends will be set to hard when VM/ESA is IPLed. Subsequent issues of the SET ABEND SNAPDUMP aaannn, where aaannn is a soft abend, will have no affect on the system.

```
set abend snapdump aas001
Ready; T=0.01/0.01 11:06:00

q abend
11:06:02 ABEND SNAPDUMP AAS001
Ready; T=0.01/0.01 11:06:02
```

Figure 89. Sample CP SET ABEND and CP QUERY ABEND Commands

For further information on the CP SET ABEND and CP QUERY ABEND commands, refer to *VM/ESA CP Command and Utility Reference*.

7.3 VM/ESA Dump Enhancements

As VM/ESA evolved, several modifications needed to be made to the dump viewing component to support the new functionality. Some of these enhancements will be detailed in this section.

7.3.1 Dump Format Changes

In previous releases of VM/ESA, the dump formats from various commands differed. As a result, different utilities had to be used to analyze these dumps.

With VM/ESA Release 2.2, dump formats have been standardized. As a result, all the following files will be in the same format:

- Virtual machine dumps
- Stand alone dumps
- CP dumps, including SNAPDUMPs, system restart dumps, and ABEND dumps
- Soft ABEND dumps.

In addition to this standardization, a new Dump File Map Block (DFMBK) is added to the dump. All blocks of data within the dump are found by fields within this map. This will allow future enhancements to take place to the dump files without introducing any compatibility issues with releases of VM/ESA from Release 2.2 onwards.

Incompatibility Issue

Because of the changes described in this section, the format of the dumps created on pre-VM/ESA Release 2.2 systems are incompatible with VM/ESA Release 2.2 DUMP processing utilities.

For further information on the CP SET ABEND and CP QUERY ABEND commands, refer to *VM/ESA CP Command and Utility Reference*.

7.4 VM Diagnostic Tools

Sometimes, IBM Service needs to ship internal tools to customers to aid in problem diagnosis after the customer calls with a problem that requires more definition or detail. This causes a delay while the customer waits to receive these diagnostic tools. To provide faster turnaround and better problem diagnosis, a set of VM diagnostic tools is shipped with VM/ESA Release 2.2.

These tools are being shipped without any warranty, expressed or implied. Refreshes may occur on release boundaries or by “Your Connection to VM” and not through the service stream.

The set of serviceability aids to be shipped in VM/ESA Release 2.2 have been shown to be of general use by IBM service personnel and designed for system programmer use. These tools are considered to be diagnostic, maintenance, and tuning aids, and as such, are not designed as programming interfaces. They are shipped as samples on the MAINT 193 minidisk. As such, all files are shipped with a filetype of SAMP*nnnn*, where *nnnn* represents the real file type.

To allow customers to modify these tools for their environment, all tools will be shipped with source code and build instructions.

A list of tools shipped with VM/ESA Release 2.2 follows:

AFTCHAIN

Used to determine what files are currently open within a virtual machine. It will also display or format the contents of active file table (AFT) entries associated with each open file.

This tool may help determine whether files are open on a particular virtual machine. The output of this command is shown in Figure 90.

```
AFT at address 00005600; File ID = MYFN     MYFT     A1

--------------------------------------------------------------------------------
```

Figure 90. Sample Output of AFTCHAIN (LIST

CMSDUMP

CMSDUMP is a package of utilities to enhance debugging of CMS or CMS applications. This package contains tools to support analysis of CMS generated VM dumps with VM/ESA DVF DUMPSCAN. The basic support provided is primarily in terms of control block analysis and mapping free storage utilization.

While in the CMSDUMP environment, the full power of DUMPSCAN, XEDIT, REXX and CMS are available as they are during any DUMPSCAN session.

FILETRAP

Generates a trace table of FILEDEF, OPEN and CLOSE activity. The data allows a user to see file activity by both a DDNAME and filename, filetype and filemode. FILETRAP may be used to identify FILEDEF clear and FILEDEF reuse problems. It will also help identify the reuse of DCBs.

FS2SFSER

Maps the contents of the DMSFSERR Table. This contains the last 11 errors that have been encountered by CMS native file system (FS) or macros calling Shared File System (SFS) functions.

This is particularly useful to customers experiencing intermittent file system related errors.

MDCHECK

This is a powerful minidisk verification and analysis tool. MDCHECK looks at a minidisk structure and analyzes block usage, flagging blocks that are invalid, not allocated, or multi-allocated. This tool gives you all the information the DFSMS CHECK function does.

MDCHECK is very useful during diagnosis of minidisk corruption. An example of MDCHECK usage is shown in Figure 91.

```
mdcheck 191

Address:   0191  Device type:      3390   Date created: 94/02/24 14:09:13
VOLID:   TIM191  Block size:       4096   Last changed: 94/03/21 10:18:09
Cyls:         5  Usable cyls:        5

Total blocks (ADTNUM):    900
Blocks used (ADTUSED):    576  ( 64%)
Blocks counted:           576
Bits set in ALLOCMAP:     576

Lost blocks:                0
Invalid blocks:             0
Multiply-used blocks:       0

Disk origin pointer:                  5
Files reported in DIRECTOR:          18  (including DIRECTOR and ALLOCMAP)
Number of files found:               18
Number of invalid FSTs:               0
Files with bad data block count:      0

Ready; T=0.01/0.01 10:18:39
```

Figure 91. Sample Use of MDCHECK

LCTRACE

This tool traces entries to and exits from the LC (Linkage Component) of CMS/SFS. The LC resides in the CMS user machine and handles communications between the user machine and the SFS server machines for all file pool requests. LCTRACE output is displayed on the user machine console, and contains input and return information for each file pool request.

PRINTBLK

This tool displays disk blocks on an accessed CMS formatted minidisk. PRINTBLK is useful for examining CMS minidisk structure by viewing its actual disk resident format and supporting hexadecimal format structures which are not accessible by conventional interfaces (for example XEDIT). Given two input parameters of the diskmode and block number, it produces output similar to the CP DISPLAY T000020.1000 command.

PRINTFST

This tool displays the formatted version of a particular file status table (FST). The tool is used to debug errors in CMS files, and may be used to display

FST information for a specified file on any accessed minidisk of a SFS directory. The output of this command is shown in Figure 92.

```
printfst fixed exec a

FST at address 00001D80 follows:

+0000  C6C9E7C5 C4404040 C5E7C5C3 40404040  * FIXED   EXEC     *
+0010  00000000 00000000 C1F10000 0000E580  *         A1    Vø *
+0020  00000042 00000000 0000100E 00000001  *    â             *
+0030  00000006 000C9404 25160621 C1        *       m     A    *

Formatted FST Follows:

FSTFILE  (file-id)      = FIXED    EXEC     A1
FSTFB    (flag byte)    =       80
FSTIL    (item length)  =       66 (00000042 hex)
FSTFV    (recfm)        =       V
FSTFOP   (file origin)  =     4110 (0000100E hex)
FSTADBC  (alt-block ct) =        1 (00000001 hex)
FSTAIC   (alt-no. items)=        6 (00000006 hex)
FSTNLVL  (no. ptrlevels)=        0
FSTREALM (real filemode)=       A

================================================================================
```

Figure 92. Sample Output of PRINTFST

MONVIEW

This tool formats, views, and manipulates CP monitor records. MONVIEW is useful to system programmers who need to analyze raw CP monitor records while doing performance tuning, performance analysis, or problem determination on products that use monitor data as input.

SFSDOT

This addition provides documentation for a set of diagnostic aids built into the Shared File System in the format of flat files which can be edited. These diagnostic aids may be issued from the SFS operator console.

Note: LCTRACE and SFSDOT are designed for use in diagnosing SFS applications and server errors respectively, under the guidance of IBM support personnel.

7.5 TRSOURCE Selectivity

In earlier releases of VM/ESA, all required data was collected if you were tracing with the TRSOURCE command option TYPE DATA.

With the new TRSOURCE selectivity enhancement, you now have the possibility to conditionally select trace events. This means when a trace point is executed, the trace can now be specified to collect data only if certain conditions exists. If the conditions do not exist, no trace record will be created. Here are the advantages that are provided by TRSOURCE selectivity.

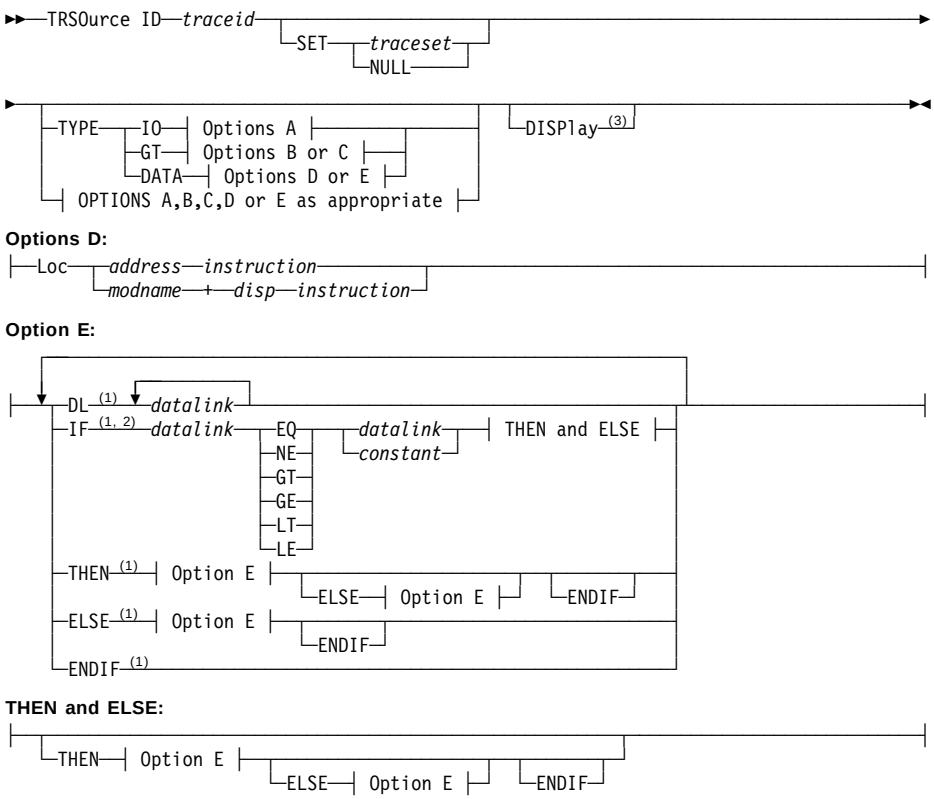
- Less data is collected. Thus, there are fewer data buffers to manage and the trace has less of an impact on the system performance.
- Only relevant data will be collected. Less filtering of the collected data is necessary.

- There is no loss of data due to “overrunning” the data saving function.

To allow the collection of selective data for data traces, the TRSOURCE command (TYPE DATA) has been enhanced. Also, the display command QUERY TRSOURCE and TRSAVE QUERY (DVF subcommand) have been modified to show the datalinks.

7.5.1 TRSOURCE Command

The selectivity of the data trace function (TYPE DATA) has been expanded to process IF, THEN, ELSE and ENDIF statements. In Figure 93 you can view a partial command syntax chart. In this figure, options A, B, and C are not shown as they were not enhanced.



Notes:

1. Use of this keyword is dependent on the keywords that have already been specified. For example, THEN cannot be specified unless an IF phrase immediately precedes it. Also, DL can not be specified if a THEN is expected.
2. IF statements may be nested up to 16 levels deep.
3. If specified, this must be the last operand entered.

Figure 93. TRSOURCE Command Syntax

DL *datalinks*

There are two new character groups that can be used to make up a character string:

- Addresses: substrings consisting of characters 0 to 9 and A to F.

- Registers: range of registers may be specified by using a colon between the registers, such as G0:4.

IF *datalinks*
Identifies the data to be used as the first operand for data comparison that determines whether a subsequent THEN or ELSE statement is executed.

THEN DL *datalinks*
Identifies the data to be traced if the result of the preceding IF statement is true. A THEN statement may be followed by any number of datalinks, all of which are executed only if the result of the preceding IF statement is true.

THEN IF
Identifies a nested IF statement. Such an IF statement would be executed only if the result of the preceding IF statement is true.

ELSE DL *datalinks*
Identifies the data to be traced if the result of the preceding IF statement is false. An ELSE statement may be followed by any number of datalinks, all of which are executed only if the result of the preceding IF statement is false.

ELSE IF
Identifies a nested IF statement. Such an IF statement would be executed only if the result of the preceding IF statement is false.

ENDIF
Identifies the end of conditional datalinks associated with the last unclosed IF statement.

Up to 16 nested IF statements are allowed. Each IF statement must be paired with an ENDIF statement.

7.5.2 QUERY TRSOURCE Command

No new options were added to this command, but the output of this command will display the datalink information in a “structured” format. To see the structured format, see the section 7.5.4, “Using TRSOURCE Command with Selectivity” section.

7.5.3 TRSAVE QUERY Subcommand

The TRSAVE QUERY is a DVF (Dump Viewing Facility) subcommand (as well as a CP command, but we are discussing the DVF subcommand here). No new options were added to this command. The output of this command has been enhanced to display the datalink information. The output is formatted in the same style as the QUERY TRSOURCE command.

7.5.4 Using TRSOURCE Command with Selectivity

This is a simple example of how to use TRSOURCE with datalinks and the IF, THEN and ELSE options. The trace of module HCPCFD (HCPCFD is the module for CP locate commands) will be activated if user STROHMAI enters the command LOCATE OPERATOR. It would be easier to put the commands in a little EXEC. This EXEC is shown in Figure 94 on page 133 and run to set up an example trace set.

```
/* */
address command
"TRSOURCE ID EXAMPLE TYPE DATA LOC HCPCFD + 1A6 58F0063C'"
"TRSOURCE ID EXAMPLE DL G0:F"
"TRSOURCE ID EXAMPLE IF GB+80.8 EQ C'STROHMAI'"
"TRSOURCE ID EXAMPLE THEN IF G8 EQ G9"
"TRSOURCE ID EXAMPLE      THEN DL G8.20 ENDIF"
"TRSOURCE ID EXAMPLE   IF G0 EQ X'00000008' THEN DL GC.30 ENDIF"
"TRSOURCE ID EXAMPLE   DL X0 G9-8.50"
"TRSOURCE ID EXAMPLE ELSE IF GC GE GB"
"TRSOURCE ID EXAMPLE      THEN IF GC LT GA"
"TRSOURCE ID EXAMPLE           THEN DL GA"
"TRSOURCE ID EXAMPLE           ELSE DL GA GB"
"TRSOURCE ID EXAMPLE ENDIF ENDIF ENDIF"
"TRSOURCE ID EXAMPLE DL G5 G5.4 G5%"
```

Figure 94. Sample REXX EXEC Using TRSOURCE

Enable the trace.

```
trsource enable id example
Ready; T=0.01/0.01 13:28:32
```

Display the TRSOURCE statements.

```
q trsource

  SET NULL

  ID       TYPE     SET      STATUS   LOC             INSTRUCTION
  EXAMPLE  DATA     NULL     ENABLED  HCPCFD   + 01A6 58F0063C0DEF

  DATALINKS
  DATALINK 001  G0:F
                IF GB+80.8 EQ C'STROHMAI' THEN
                 IF G8 EQ G9 THEN
  DATALINK 002    G8.20
                 ENDIF
                 IF G0 EQ X'00000008' THEN
  DATALINK 003    GC.30
                 ENDIF
  DATALINK 004   X0
  DATALINK 005   G9-8.50
                ELSE
                 IF GC GE GB THEN
                  IF GC LT GA THEN
  DATALINK 006     GA
                   ELSE
  DATALINK 007     GA
  DATALINK 008     GB
                  ENDIF
                 ENDIF
                ENDIF
  DATALINK 009  G5
  DATALINK 010  G5.4
  DATALINK 011  G5%
  Ready; T=0.01/0.01 16:49:34
```

Trigger the trace by using the LOCATE command.

```
locate operator
  VMDBK    = 3D3A4000
  Ready; T=0.01/0.01 13:31:01
```

Disable and drop the trace, then process the trace.

```
trsource drop id example
  Ready; T=0.01/0.01 13:31:14

q trf
  OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
  STROHMAI 0029 TRF  A  0001 03/16 13:30:50 EXAMPLE  DATA     SYSTEM
  Ready; T=0.01/0.01 13:31:40

tracered 0029
  HCSTSS083I PROCESSING SYSTEM TRACE FILE(S) FOR 0029 CREATED 03/16 13:31:47
  HCSTSP010A ENTER TRACE ENTRY SELECTION CRITERIA, NULL LINE TO END SELECTION, O
  QUIT TO END TRACERED COMMAND
  HCSTSM084I PROCESSING COMPLETE - 1 TRACE ENTRIES FORMATTED
  PRT FILE 0031 SENT FROM STROHMAI PRT WAS 0031 RECS 0030 CPY  001 A NOHOLD NOKE
  Ready; T=0.01/0.02 13:34:12
```

The contents of the trace are shown in Figure 95.

```
 -------------------------- 03/14/94 18:01:47.021263 --------------------------


TRACE TYPE DATA, CPU 0000                TIME 18:01:47.021263
      TRACEID = EXAMPLE, TRACESET = NULL, ADDRESS = 00219606
   -> DATALINK STRING   1 = G0:F
      DATA = 00000008 3D3A4000 002194E6 3D3A4000  *...... ...mW.. .*
             E3C54040 00219488 00000000 00000000  *TE  ..mh........*
             373B9D10 003F3F08 3D275E78 37009000  *..........;.....*
             00219460 3E315A80 802195EE 00207C48  *..m-..!...n...@.*
   -> DATALINK STRING   2 = GC.30
      DATA = 47F0F028 C8C3D7C3 C6C44040 00FAC5E2  *.00.HCPCFD  ..ES*
             C1F2F2F0 0000F0F1 61F0F761 F9F47AF0  *A220..01/07/94:0*
             F24BF4F3 D7C50000 5FC0C730 41000008  *2.43PE..¬.G.....*
   -> DATALINK STRING   3 = X0
      DATA = 14B1EE40                             *... *
   -> DATALINK STRING   4 = G9-8.50
      DATA = 6E6E0005 80132E40 00000000 00000000  *>>..... ........*
             80080005 000F000F D3D6C3C1 E3C540D6  *........LOCATE O*
             D7C5D9C1 E3D6D915 00000000 00000000  *PERATOR.........*
             4CC7E2C4 4C4C4C4C 800001C8 00C3D5C6  *<GSD<<<<...H.CNF*
             6E6E0005 80132E40 003F3DC8 19FF4000  *>>..... ...H.. .*
   -> DATALINK STRING   5 = G5
      DATA = 00219488                             *..mh*
   -> DATALINK STRING   6 = G5.4
      DATA = 5FC0C730                             *¬.G.*
   -> DATALINK STRING   7 = G5%
      ********** ERROR - INVALID LOCATION **********
```

Figure 95. TRSOURCE Trace Output

Chapter 8. Hardware Support

Since VM/ESA can be used either to support production users or provide the facilities to efficiently run several second-level operating systems, timely support of new hardware is of major importance. This allows the users of VM/ESA to maximize the benefit they receive from running VM/ESA.

This chapter describes several hardware and guest system enhancements.

8.1 Integrated Console Support

Integrated console support allows VM to communicate with the master console on the ES/9000 series of mainframes. This allows the customer to use the service console as an interface to VM/ESA, as opposed to a dedicated 3270 screen.

In previous releases of VM/ESA running on ES/9000 hardware, the system consoles had to be locally attached to allow CP to initialize. If the master or alternate consoles were not available to VM, CP would load a X'1010' PSW wait-state and the IPL of VM would be aborted. The main objective of this enhancement allows CP to be initialized when a locally attached 3270 operator's console is not available.

The integrated console can be set up so that it is always the master console for the system, or it can be used for disaster recovery. These options can be activated by:

- Changing the SYSTEM CONFIG to incorporate a permanent change using the SYSTEM_CONSOLE statement in place of a hardware address
- Changing the SAPL screen at IPL time to incorporate a temporary change using CONS=SYSC, where SYSC is the logical address of the hardware console.

The following sections describe both of these options.

Full-Screen Support

The system console provides a line-mode interface to the operator. As a result, full-screen applications may not run in the environment. One of the most important consequences of this is the stand-alone program loader (SAPL), a full screen application. For disaster recovery purposes, the System_Console operand should be placed at the end of consoles defined in the *Operator_Console* macro of the SYSTEM CONFIG file.

If you lose all your local 3270 VM system consoles, your VM system can still be IPLed. If you have not added this option to the SYSTEM CONFIG and you lose all your locally attached VM system consoles, you will be unable to IPL the SAPL to dynamically define the CONS=SYSC, as shown in Figure 97 on page 137.

The operator console for your mainframe may not perform as well as your normal 3270 VM console, however, in an emergency it may decrease the downtime of your VM system.

8.1.1 System Configuration Changes

To make the system console of your ES/9000 mainframe the operator console for your VM/ESA Release 2.2 machine, you must add a new operand to the system configuration file. This new operand can be added to the *Operator_Console* macro. This will ensure that all messages will go to the OPRMSG panel of your ES/9000 mainframe.

Should you only require emergency messages to go to your system console, you can add the operand to the *Emergency_Messages_Console* macro within the SYSTEM CONFIG File.

If you already have a good running 3270 display used for your VM operator console, you may want to add the integrated hardware console as the last console defined. That way, if all terminal control units are unavailable, you still have a way to IPL. Figure 96 shows an example of a SYSTEM CONFIG file coded so that the VM master console and emergency messages will be sent to a series of local screens, then finally the ES/9000 system console.

```
/*********************************************************************/
/*                     Console Definitions                           */
/*********************************************************************/

 Operator_Consoles            0020 0021 08E0 0840 0E20 0E21 1020,
                              System_Console
 Emergency_Message_Consoles   0020 0021 08E0 0840 0E20 0E21 1020,
                              System_Console
```

Figure 96. Sample SYSTEM CONFIG for Integrated Console Support

8.1.2 Testing the Integrated Hardware Console Facility

To make yourself and the system operators more familiar with this new type of console interaction, you should do some testing. This can be done with a temporary override on the SAPL panel. IPL with a loadparm pointing to a local 3270 on which the SAPL menu will appear, as shown in Figure 97. Specify the SYSC operand in the CONS statement of the SAPL menu, as shown, and all the messages will appear on your mainframe's system console. A sample IPL is shown in Figure 103 on page 153.

```
STAND ALONE PROGRAM LOADER: VM/ESA RELEASE 2.2

DEVICE NUMBER:   01FF       MINIDISK OFFSET:   00000000   EXTENT:  -

MODULE NAME:     CPLOAD     LOAD ORIGIN:       1000

--------------------------------IPL PARAMETERS-----------------------
cons=sysc


----------------------------------COMMENTS--------------------------




------------------------------------------------------------------------

9= FILELIST  10= LOAD  11= TOGGLE EXTENT/OFFSET
```

Figure 97. Sample SAPL Panel for Integrated Console Support

8.1.3 Modified Command Outputs and Diagnose Codes

Since the mainframe system console does not have a physical screen address, the output of some commands and diagnose codes have changed. These changes will be detailed in this section.

DIAGNOSE Code X'24'

When DIAGNOSE Code X'24' (Device Type and Features) is issued for the virtual console and the user ID is running on the mainframe system console, CP will return the information that indicates the real device is an undefined line-mode terminal.

This maintains compatibility with applications that currently use DIAG X'24'. For further details on DIAG X'24', refer to *VM/ESA CP Programming Services*.

DIAGNOSE Code X'210'

Like DIAG X'24', when DIAGNOSE Code X'210' (Retrieve Device Information) is issued for the virtual console and the user ID is running on the mainframe system console, CP will return the information that indicates the real device is an undefined line-mode terminal.

This maintains compatibility with applications that currently use DIAG X'210'. For further details on DIAG X'210', refer to *VM/ESA CP Programming Services*.

LOGON/DISCONNECT Commands

If the user ID has been logged on or disconnected from the master console of an ES/9000 processor, where normally you would expect to see the screen address of a 3270 screen, you will now get *SYSC*. An example of this is shown in Figure 98.

```
DISCONNECTED FROM SYSC
```

Figure 98. Sample DISCONNECT from User ID on the Mainframe System Console

For additional information about these commands, see *VM/ESA CP Command and Utility Reference*.

QUERY USERID/QUERY CONSOLE/XAUTOLOG

These commands behave in the same way as described in section "LOGON/DISCONNECT Commands" on page 137. In the output of these commands where you would expect to see a screen address, if the user ID is logged on at the mainframe system console, you will see *SYSC*.

For additional information about these commands, see *VM/ESA CP Command and Utility Reference*.

8.1.4 Sample IPL at ES/9000 System Console

Appendix C, "VM/ESA Release 2.2 IPL Using Integrated Console Facility" on page 153, shows a sample IPL using the mainframe system console.

ES/9221 Processor Support

As the ES/9221 hardware console uses a graphical user interface (GUI), the messages produced on the console will appear in a window, as distinct from the OPRMSG panel on the ES/9121 and ES/9021 processors.

8.1.5 Running VM/ESA Release 2.2 Second-Level

To test this feature, you may want to run your VM/ESA Release 2.2 system second-level. The integrated console is supported second-level. To issue commands to the second-level VM/ESA Release 2.2, you should use the CP VINPUT command. Use VINPUT to request the VM/ESA control program to send the accompanying data to the operating system running in your virtual machine. To the operating system, the data appears to have been entered from the system console. Using SCIF (CP SEND) and VINPUT combined with hardware console support, there is the possibility to further automate the IPL of a second-level VM system.

For more information about the VINPUT and SEND commands, refer to *VM/ESA CP Command and Utility Reference*.

8.2 3990 Model 6 Support

The new 3990 Model 6 control unit is now supported in VM/ESA Release 2.2. Support of 3990 Model 6 uses existing error conditions, recovery, and messages currently used for prior 3990 models.

Functional Changes:

- Storage path status now returns a 24-byte record information, which includes the number of logical paths fenced. The QUERY FENCES command will be updated.
- Support of PSF suborders X′B0′ (Set Interface ID).
- A new suborder, X′0B′ logical path status, has been added. It returns the status of all logical paths in the subsystem. Model 6 support expands the number of logical paths from 16 to 128.
- Two new Quiesce/Resume channel path functions:
  - Quiesce an interface: The purpose is to stop all activity on an ESCON interface (or port). This is intended to be used for two types of maintenance:
    - ESCON channel wraps
    - Microcode update.
  - Quiesce a Storage Cluster: The purpose is to stop all activity on the storage controller. The support facility will ask each host to stop using the channel paths to that storage cluster.

  VM/ESA Release 2.2 is functionally updated to handle messages and Quiesce/Resume responses.

8.2.1 DASD Control Levels

When you attach or dedicate devices whose controllers have advanced features, it is important to be aware that many of the advanced features are available only if the user has the DEVCTL or SYSCTL directory options. In Table 10, these features and their corresponding options are listed for you.

CU Type	Additional Control CCWs accepted for DEVCTL	Additional Control CCWs accepted for SYSCTL
3990-3 3990-6	• Set subsystem mode – Activate caching for device – Deactivate caching for device – Activate DASD fast write – Deactivate DASD fast write – Force deactivate DASD fast write • Perform subsystem function – Establish duplex pair – Terminate duplex pair – Suspend duplex pair – Direct I/O to one device of the duplex pair	• Set subsystem mode – Make cache available – Make cache unavailable – Force cache unavailable – Make NVS available – Make NVS unavailable – Activate cache fast write – Deactivate cache fast write • Perform subsystem function – De-stage modified tracks

Table 10. DASD Control Levels

8.2.2 Control Unit Initiated Reconfiguration (CUIR)

The Model 6 supports the new CUIR feature. This feature allows a service representative (SR) to initiate reconfiguration requests from the control unit.

Currently, if a SR has to do service maintenance at a device, the operator has to vary offline all paths to that device and the device itself has to be set offline. In a multiple processor environment, you have to know the exact configuration in order to set all required paths offline. With the new CUIR feature, channel paths can be quiesced from the control unit, which will inform all attached processors that specified channel paths will be made unavailable and requests that these processors stop using those channel paths. Refer to Appendix D, "QUIESCE/RESUME Request with a 3990 Model 6" on page 155.

8.2.3 New CP Command Responses

Shown in Figure 99 are several CP command outputs which show the quiesced paths possible with the 3990 Model 6 controller.

```
q dasd details db3
  ODB3  CUTYPE = 3990-CC, DEVTYPE = 3390-0A, VOLSER = TOTE22
        CACHE DETAILS:  CACHE NVS CFW DFW PINNED CONCOPY
             -SUBSYSTEM   M    T   Y   -    N       N
                -DEVICE   Y    -   -   N    N       N
        DEVICE DETAILS: CCA = B3, DDC = 33
        DUPLEX DETAILS: SIMPLEX
  Ready; T=0.01/0.01 14:14:17

q paths db3
  DASD ODB3 online
  Path status for device ODB3: online - 1B 1F, offline - 10 14
                               quiesced - 20 24
  Ready; T=0.01/0.01 14:15:25

query chpid 1B
  Path 1B quiesced to devices ODB0 ODB1 ODB2 ODB3
  Ready; T=0.01/0.01 14:18:11
```

Figure 99. New Response to CP Q DASD DETAILS Command

For subsystem CACHE, the status can be one of the following:

Y subsystem caching active
N subsystem caching inactive
T subsystem caching terminated
M subsystem caching disabled for maintenance
F subsystem caching pending inactive, de-stage has failed

For subsystem NVS, the status can be one of the following:

Y subsystem NVS active
N subsystem NVS inactive
T subsystem NVS terminated
M subsystem NVS disabled for maintenance

8.2.4 3990 Migration

There are data migration considerations to take into account if migrating from any existing 3990 to a control unit 3990 Model 6. Model 6 control units will only support 3390 devices. All existing 3380 devices are not supported. Also, 3380 track compatibility mode on 3390 DASD is not supported. 3390 DASD attached to a Model 6 control unit will only be allowed to operate in native mode.

8.2.5 Performance

The 3990 Model 6 supports channel data rates up to 4.5 MB/Sec on parallel interfaces and up to 18.18 MB/Sec on serial interfaces. The cache of a Model 6 has a maximum data rate of 120 MB/Sec, so up to four channel interfaces (ESCON) can transfer 18.18 MB/Sec at a time without exceeding cache bandwidth.

8.3 Processor Support

It is still a key factor in VM/ESA development to support the latest processors. Below is a list of the latest processor support.

8.3.1 9021 Support

9021 Model 832

CP currently supports a three-processor 9021 (Model 831) and also partitionable 9021 processors (many models).

Now VM/ESA 2.2 supports the new 9021 Model 832, which is a three-processor machine that is partitionable into a one-way and a two-way processor.

9021 Model 9X2

The 9021 Model 9X2 is a 10-CPU (5+5) machine, making it the world's most powerful general computing processor.

VM/ESA will support this processor native, in single image, with no design changes.

8.3.2 9221 Support

VM/ESA Release 2.2 supports the new rack-mounted 9221 processors:

- Model 191
- Model 201
- Model 211
- Model 221
- Model 421

8.3.3 New Model Dependent RC for Read IOCDS Function

CP has been enhanced to support a new return code (RC) from a future level of service processor microcode for the 9021 and 9121 processors. Affected VM systems will not be able to execute a READ-IOCDS operation without this enhancement.

Appendix A. VM Open System Statement of Direction

This appendix provides details of the statements of direction issued by IBM, with reference to POSIX and DCE support under the VM/ESA platform.

A.1 Statement of General Direction - DCE on VM/ESA

IBM′s Open Systems Strategy positions VM to play a key role in heterogeneous, multi-vendor environments by providing the infrastructure on which solutions are implemented to meet our customers′ business needs. These environments include hardware and software, from both IBM and OEM vendors, encompassing mainframes, intelligent workstations, terminals, and other peripheral devices. Interoperability and portability are key elements of this infrastructure.

Interoperability is enabled in VM/ESA environments with the support for industry standard protocols for communications across local area networks (LANs) and wide area networks (WANs). In addition to the basic communications protocols, VM will support industry standard services, such as the Open Software Foundation** (OSF**) Distributed Computing Environment (DCE), where these services fill functional needs in our customers′ environments.

On February 9th, 1993, and again on May 20th, 1993, IBM announced the following Statement of Direction (SOD) stating the intent to support the Open Software Foundation**′s Distributed Computing Environment (OSF** DCE) on the VM/ESA operating system platform:

> DCE is a collection of technologies selected by the OSF** based on industry acceptance and technical merit. The purpose of DCE is to support fully integrated distributed computing in a heterogeneous multi-vendor environment. In the complex, rapidly changing world of open systems, the DCE is an infrastructure on which solutions can be implemented to meet our customers′ business needs. It is designed to address today′s challenges and accommodate new technologies of the future.
>
> To further address market requirements regarding the open systems environment, IBM has added plans for the introduction of DCE support for VM/ESA. IBM intends to support selected components of the OSF** DCE as client/server enabling services on the VM/ESA platform. VM/ESA′s support of OSF** DCE will allow VM applications to participate in the DCE environment. Each of the IBM supported DCE implementations will interoperate with other DCE-compliant systems. OSF** DCE plays a key role on each of the IBM systems, enabling enterprise integration for our customers by providing interoperability across complex, distributed computing environments.

In the announcement letter for VM/ESA Release 2.2, additional information about IBM′s plans on implementing DCE on the VM/ESA operating system platform were provided.

The DCE support on VM/ESA will enable VM to participate in a DCE cell and support DCE Remote Procedure Call (RPC) based client and server applications. VM support will be directed at components of the DCE Executive. It is IBM′s

intent for VM/ESA to support all of the components of the DCE Executive in the future.

The DCE Executive consists of the following components:

- DCE Remote Procedure Call (RPC)

  VM/ESA will provide the complete structure for connection-less RPC. RPC provides a facility for calling a procedure on a remote system as though it were a procedure on the local system. RPC also provides a high-level programming model which masks the application from the underlying details of the communications network.

- DCE Threads

  VM/ESA will provide support for the DCE Threads application programming interface (API). The DCE Threads service allows a user to create and control multiple threads of execution within a single process. The user threads library implementation is based on IEEE POSIX 1003.1C.

- DCE Cell Directory Service (CDS) Client

  The DCE CDS Client provides access to the DCE CDS Server on another system in the DCE Cell.[1] The DCE Directory Services provide consistent naming throughout the distributed environment. This allows users to identify resources, such as RPC-based servers, files, or print queues, by name and gain access to them without needing to know their location in the network.

- DCE Security Service Client

  The DCE Security Service Client provides the API to access the DCE Security Server on another system in the DCE Cell. The DCE Security Server provides user registration, authorization, and authentication services for distributed applications.

- DCE Distributed Time Service (DTS) Server and Client

  The DCE DTS provides time services that enable distributed applications to determine event sequencing, duration, and scheduling.

A.2 Statement of General Direction - POSIX on VM/ESA

On February 9th, 1993, and again on May 20th, 1993, IBM announced the following Statement of Direction (SOD) stating the intent to support selected Institute of Electrical and Electronic Engineers (IEEE) POSIX standards on the VM/ESA operating system platform:

> The Institute of Electrical and Electronic Engineers (IEEE) Technical Committee on Open Systems (TCOS) has defined a base set of operating system services which are intended to provide greater system consistency across unlike operating environments. These services are formally defined in a set of standards known as POSIX, a Portable Operating System Interface for Computer Environments. They were developed from an open forum by technical professionals from many companies, including IBM. The implementation of POSIX services on IBM operating system platforms will enhance our customers′ ability to

[1] A DCE Cell is an administrative customer-defined domain that consists of a set of network-connected client and server nodes. A DCE Cell requires at least one DCE Cell Directory Server and one DCE Security Server.

interoperate with operating environments from other vendors and to support POSIX-compliant applications.

To further address market requirements regarding the open systems environment, IBM has added plans for the introduction of selected POSIX standards on VM/ESA. IBM intends to support IEEE 1003.1 (System Interfaces) and applicable Federal Information Process Standard (FIPS 151), and IEEE 1003.1C (Threading) POSIX standards on VM/ESA. As additional POSIX standards and other worldwide open standards become formally approved, IBM will evaluate the applicability of those standards on VM/ESA to satisfy customers′ requirements, and will implement them as appropriate.

In the announcement letter for VM/ESA Release 2.2, IBM announced that it is adding IEEE 1003.2 Shell and Utilities to the suite of POSIX standards it intends to support on VM/ESA. The POSIX Shell and Utilities will provide a UNIX** style application development environment on VM. It will also provide application services on VM for the development of POSIX based applications.

A.3 CMS Graphical User Interface

The Open Software Foundation**′s Motif** provides a window management and presentation system that presents users with a windowing environment that can be integrated with their existing workstations. The X Window System and Motif** are currently available for use by applications on VM/ESA with the IBM Transmission Control Protocol/Internet Protocol (TCP/IP) program product. The TCP/IP interface provides a solution which exists today, and is cross platform portable. It is useful when porting X Window applications to VM and using VM to develop applications which are required to run on other platforms.

However, consider the following challenges:

- TCP/IP interface only. VM has a large customer base on SNA.
- Requires customers to purchase X Servers, an expensive pre-requisite for many to add this support.
- Interface to workstation is not optimized for networking that will be typical for mainframe users.

With these challenges in mind, VM planning and development put together a set of goals for an IBM developed GUI API to be shipped with VM. The list of goals is as follows:

- The new GUI interface should have a flexible API
  - Based on object oriented principles. This will be more compatible with the current desktop environments. Also, an application on VM that handles the manipulation of objects rather than lower-level concepts, such as drawing, is better suited to a Client/Server mainframe application.
  - Applications should be able to access this interface from as many supported languages as possible. For VM, this means having a CSL interface. Plans are to provide an API callable from C, C++, and REXX in a form that is most natural to that language.
- A CMS desktop should allow users to logically separate host functions from their 3270-emulator window. For example, to perform VM reader functions,

you can use RDRLIST in the 3270 window or use an icon on the desktop independent of the 3270 window.

- You should be able to run multiple applications using this interface on CMS. For example, you can have open windows to do things like RDR management or file management on the desktop, and work in the 3270 window as well.
- With regard to the workstation, the native workstation OS/GUI should be used. This will mean that an OS/2 user will see a CMS application running in an OS/2 window. Certain OS/2 desktop functions could be performed against the CMS desktop application, such as using the OS/2 paint brush to change a background color.
- The workstation code should be dynamically versioned. When the code on your workstation connects to VM, the host will detect if the level of the code on your workstation is at the same level as the host code. If not, the host will down-load the proper level before proceeding.
- Access to VM should be provided without requiring you to explicitly log on. An icon on the desktop could be a VM application that when “clicked-on” would log onto VM (if not already logged on) and execute the application.
- A user should be able to use their PC-based editor against a host file and also use XEDIT on a PC file.
- XEDIT should take advantage of the extended functions in the workstation environment but retain the current look and feel of XEDIT.
- This new interface should work over both TCP/IP and SNA.
- All these items should be provided with no additional cost to their system, both in real dollars and with little or no degradation to the performance.

IBM will provide prototypes to demonstrate to VM/ESA customers the use of GUI applications using VM, and to allow customers to comment to IBM on the applicability and usefulness of these types of applications on VM. These prototypes will be made available during 1994.

Appendix B. Local Modification to SYSPROF EXEC

This appendix provides details on how to make a local modification to your system's SYSPROF EXEC. This modification will be logged in the VMSES/E maintained software inventory. As a result, should you receive any corrective service for the SYSPROF EXEC, at the end of the apply stage you will be notified that your local modification may need reworking.

B.1 Local Modification Application Procedure

To easily show our modification, we have added a line to the SYSPROF EXEC which displays the message shown below. The message is displayed immediately after the execution of the user's PROFILE EXEC, by processing within the SYSPROF EXEC.

```
This is our local mod to the SYSPROF EXEC
```

The procedure for adding a local modification to a source maintained file (that is an EXEC or XEDIT macro) is detailed in Chapter 5 of *VM/ESA Service Guide*. This procedure was followed during this local modification application.

The steps to apply the local modification are:

1. Create or update the AUXLCL file for the SYSPROF EXEC.
2. Create an update file for SYSPROF EXEC.
3. Rebuild the source file with the local modification applied.
4. Copy the new SYSPROF EXEC to the CMS System Disk.
5. Re-save the CMS saved segment.
6. Re-save the CMSINST saved segment.
7. Test your local modification.

If you are unfamiliar with the concepts of local AUX files (AUXLCL) and local Version Vector tables (VVTLCL), you should consult *VM/ESA VMSES/E Introduction and Reference* and *ITSO VMSES/E Primer: Concepts and Experiences*.

Access CMS Minidisks

Setup the CMS minidisks (if you have not already done so).

```
vmfsetup esa cms
VMFSET2760I VMFSETUP processing started for ESA CMS
VMFUTL2205I Minidisk|Directory Assignments:
            String    Mode  Stat  Vdev  Label/Directory
VMFUTL2205I LOCALMOD  E      R/W  3C4   MNT3C4
VMFUTL2205I LOCALSAM  F      R/W  3C2   MNT3C2
VMFUTL2205I APPLY     G      R/W  3A6   MNT3A6
VMFUTL2205I           H      R/W  3A4   MNT3A4
VMFUTL2205I           I      R/W  3A2   MNT3A2
VMFUTL2205I DELTA     J      R/W  3D2   MNT3D2
VMFUTL2205I BUILD7    K      R/W  493   MNT493
VMFUTL2205I BUILD6    L      R/W  490   MNT490
VMFUTL2205I BUILD5    M      R/O  19D   MNT19D
VMFUTL2205I BUILD2    N      R/W  193   MNT193
```

```
VMFUTL2205I BASE2     0      R/W  3B2   MNT3B2
VMFUTL2205I BASE3     P      R/W  393   MNT393
VMFUTL2205I --------  A      R/W  191   MNT191
VMFUTL2205I --------  B      R/W  5E5   MNT5E5
VMFUTL2205I --------  D      R/W  51D   MNT51D
VMFUTL2205I --------  S      R/O  190   MNT190
VMFUTL2205I --------  Y/S    R/O  19E   Y-DISK
VMFSET2760I VMFSETUP processing completed successfully
Ready; T=0.67/0.72 16:58:39
```

Create or Update the AUXLCL File for the SYSPROF EXEC

The AUXLCL file defines the sequence by which local modifications are applied to the system. The first update is taken from the bottom of the AUXLCL file and the last from the top. If you already have an AUXLCL file for your SYSPROF EXEC, this latest update should be placed at the top of the file.

For this test, we have no previous service to the SYSPROF EXEC, and as a result, we must create an AUXLCL file.

Local service files should not be placed on the same disks as the normal IBM service files. Each component of VM/ESA has a local service disk. All files created for this local modification to CMS should be placed on the MAINT 3C4 disk.

The first step we perform is to add a record to the SYSPROF AUXLCL. The AUXLCL line entry should conform to the following syntax:

updateft svclvl **lc***modid comment*

where:

updateft is the filetype of the file that will contain the local modification.

svclvl is a service level indicator. This must be in uppercase. For local modifications, this is usually LCL.

lc*modid* is **lc** concatenated with *modid*, where *modid* is the local modification number. It should begin with L and be followed by up to 4 alphanumeric characters that identify the local modification.

comment is a comment that will identify the local fix.

Figure 100 is the sample AUXLCL file we created on our E-disk for this test.

```
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
00001 LCL00001 LCL LCL0001 *  LOCAL MOD TO SYSPROF EXEC
```

Figure 100. Sample SYSPROF AUXLCL File

Create Update File for SYSPROF EXEC

The next step in the process is to create the local modification itself.

Copy the SYSPROF $EXEC to your 3C4 minidisk.

copy sysprof $exec fm-393 **= EXEC** fm-3C4 **(oldd**

Now, you XEDIT the file using the following command, and make your changes to the SYSPROF EXEC.

xedit sysprof exec fm-3C4 **(ctl dmsvm**

Once you have finished making your changes, save the changes and exit the XEDIT session. The SYSPROF LCL00001 will now be on your A disk. A sample copy of this file is shown in Figure 101.

```
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
00001 ./ I 04970000          $ 4975000 5000                 02/28/94 16:39:1
00002   SAY 'This is our local mod to the SYSPROF EXEC'
```

Figure 101. Sample SYSPROF LCL00001 File for Local Modification

Rebuild Source File with Local Modification Applied

To rebuild the SYSPROF EXEC, we use the VMSES/E VMFEXUPD command. This command was described in section 4.1, “New VMSES/E EXEC Update Function” on page 92. Several options are available for this command.

In this example, we want to rebuild the SYSPROF EXEC using the following logic:

1. Check that the VVTLCL and AUXLCL match.
2. If they do, build the new EXEC with a filetype of EXEC.
3. Store the output on the E disk, which in this case is MAINT's 3C4.

vmfexupd sysprof exec esa cms (ckgen filetype exec outmod e

```
VMFEXU2760I VMFEXUPD processing started
VMFSIP2468E Mismatch, VVT file 6VMVMA22 VVTLCL not found on a LOCAL or APPLY
            disk. Aux file SYSPROF AUXLCL E contains updates for file SYSPROF
            EXC
=========================SYSPROF EXC=========================================
6VMVMA22 VVTLCL.............| |SYSPROF AUXLCL E
___________________________|_|_____________________________________________
.........................._|X|LCL00001 LCL LC00001 * LOCAL MOD TO SYSPROF EXEC
=============================================================================
VMFEXU1965E The command, VMFSIM CHKLVL, failed with return code 8 when issued
            with the argument(s): ESA CMS
VMFEXU2760I VMFEXUPD processing completed unsuccessfully
Ready(00100); T=1.41/1.46 16:51:49
```

In this case, the command failed because the VVTLCL for CMS (6VMVMA22 VVTLCL) did not contain an update for our local modification.

In the command below, we will issue the LOGMOD option. This will update CMS's VVTLCL with information regarding our local modification. It will then create the updated SYSPROF EXEC.

vmfexupd sysprof exec esa cms (logmod filetype exec outmod e

```
VMFEXU2760I VMFEXUPD processing started
DMSUPD178I Updating SYSPROF $EXEC A1
DMSUPD178I Applying SYSPROF LCL00001 E1
VMFSIP2509I The version vector table 6VMVMA22 VVTLCL E will be updated for the
```

```
            part SYSPROF EXC using the information in AUX file SYSPROF AUXLCL
VMFEXU2507I SYSPROF EXEC created on your E-disk for use in a VMSES/E environment
VMFEXU2760I VMFEXUPD processing completed successfully
Ready; T=1.58/1.65 16:52:16
```

As can be seen above, the command completed successfully. We will now rerun the first command to show that the CMS VVTLCL has been created.

```
vmfexupd sysprof exec esa cms (ckgen filetype exec outmod e

VMFEXU2760I VMFEXUPD processing started
VMFSIP2472I VVT and AUX levels match for SYSPROF EXC
DMSUPD178I Updating SYSPROF $EXEC A1
DMSUPD178I Applying SYSPROF LCL00001 E1
VMFEXU2507I SYSPROF EXEC created on your E-disk for use in a VMSES/E environment
VMFEXU2760I VMFEXUPD processing completed successfully
Ready; T=1.50/1.57 16:52:33
```

If using the command above, the new SYSPROF EXEC is automatically created on the 3C4 minidisk. As stated earlier, this disk is used for holding local modification information for the CMS component of VM/ESA.

Test Local Modification

As we do not want to endanger the production CMS environment, we need to make a private test of this modification. VM's setup provides the MAINT 490 disk as a CMS test resident. The new EXEC should be copied there. But, copying the modified SYSPROF may not be enough, as the SYSPROF is one of the EXECs that exist in the CMS installation shared segment (CMSINST). We can do our tests with the following commands:

```
access 490 Z
copy sysprof exec fm-3C4 = = z (olddate replace
detach 190
define 490 190
define storage 17M

ipl 190 PARM INSTSEG NO
VM/ESA REL. 2.2 02/01/94 11:19

This is our local mod to the SYSPROF EXEC
Ready; T=0.03/0.04 17:10:27
```

Figure 102. Sample CMS IPL, Including SYSPROF Local Modification

Copy the New SYSPROF EXEC to the CMS System Disk

Now that we have created the modified SYSPROF EXEC, we need to copy it to the system disk so that it is available to all users. To do this, we need write access to the 190 disk. The 490 disk should also receive the update, if you have not already copied it here while testing. The procedure is shown below:

```
access 190 z
access 490 x

copy sysprof exec fm-3C4 = = z (olddate replace
copy sysprof exec fm-3C4 = = x (olddate replace
```

It is not recommended that the executable parts remain on the LOCALMOD disk. It is recommended that you rename the newly created SYSPROF EXEC on the LOCALMOD (*fm-3C4*) disk.

Re-save the CMS Saved Segment

Since we have modified the system disk, we need to re-save the CMS saved segment. This can be achieved by issuing the following commands:

```
sampnss cms
```

This creates a skeleton saved segment that will be used to save CMS.

```
ipl 190 clear parm savesys cms
```

This saves a copy of CMS in the skeleton saved segment.

Re-save the CMSINST Saved Segment

To increase the speed of operation of some commonly used EXECs, they have been saved in a saved segment so that one copy is available to all the users on the system. SYSPROF EXEC, one of these EXECs, resides in the CMSINST saved segment. This segment resides in the same segment space as the HELP saved segment. The segment space is called HELPINST.

Since we have modified the SYSPROF EXEC, we need to save the latest copy of it in the HELPINST segment space. If you are unfamiliar with concepts of saved segments and segment spaces, please refer to *VM/ESA Planning and Administration*.

The procedure for saving the HELPINST saved segment is described in Chapter 3 Step 12 of *VM/ESA Service Guide*. We will describe the steps here:

```
set machine esa
System Reset.
System = ESA

define storage 64M
STORAGE =      64M

ipl 190 clear parm nosprof instseg no mtseg no

VM/ESA Rel. 2.2 mm/dd/yy hh:mm

               ** DO NOT press ENTER **
access (noprof

access 5e5 b

access 51d d

vmfbld ppf segbld esasegs segblist helpinst (all
VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I New build requirements identified
VMFBLD1851I (1 of 1) VMFBDSEG processing SEGBLIST EXC00000, target is BUILD 51D
            (D)
VMFBDS2115I Validating segment HELPINST
VMFBDS2002I A DEFSEG command will be issued for 1 segment(s).
VMFBDS2219I Processing object HELPINST.SEGMENT
```

```
DMSACP726I 19D M released
HCPNSS440I Saved segment HELPINST was successfully saved in fileid 0145.
VMFBLD1851I (1 of 1) VMFBDSEG completed with return code 0
VMFBLD2180I There are 0 build requirements remaining
VMFBLD2760I VMFBLD processing completed successfully
Ready; T=8.33/8.75 17:09:20
```

The HELPINST segment has now been saved, and the service of the SYSPROF EXEC is completed.

Appendix C. VM/ESA Release 2.2 IPL Using Integrated Console Facility

This Appendix shows a sample IPL of VM/ESA Release 2.2 on a 9021 mainframe using the integrated console facility. The SAPL panel, as shown in Figure 97 on page 137, was used to IPL the system. The output is shown in Figure 103.

```
                                                          dd mmm yy hh:mm:ss
                     Operator Messages - Nonpriority            (OPRMSG)
Refresh (Sec):  3       Right: Priority        Other Partitions: Nonpriority
  09:45:07 VM/ENTERPRISE SYSTEMS ARCHITECTURE   RELEASE 2.2 SERVICE LEVE9401;
  09:45:07 SYSTEM NUCLEUS CREATED ON 02/01/94 AT 11:13:07, LOADED FROM TE22
  09:45:07
  09:45:07 ************************************************************
  09:45:07 * LICENSED MATERIALS - PROPERTY OF IBM*                      *
  09:45:07 *                                                           *
  09:45:07 * 5684-112 (C) COPYRIGHT IBM CORP. 1983, 1994. ALL RIGHTS    *
  09:45:07 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,     *
  09:45:07 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE   *
  09:45:07 * CONTRACT WITH IBM CORP.                                    *
  09:45:07 *                                                           *
  09:45:07 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.           *
  09:45:07 ************************************************************
  09:45:07
  09:45:07 HCPZCO6718I Using parm disk 1 on volume TOTE22 (device 0DB3).
  09:45:07 HCPZCO6718I Parm disk resides on cylinders 224 through 240.
  09:45:07
  09:45:07 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
  09:45:07       (NOAUTOlog)) or (SHUTDOWN)

SCP COMMAND ==> WARM_____________________________________


COMMAND ==>
```

Figure 103. Sample IPL of VM/ESA Release 2.2 Using Integrated Console Facility

For more information on the OPRMSG panel, and also on the OPRSUM panel which is used if your mainframe complex runs many LPARs, please refer to the Operating Manual for your specific processor.

As shown in this figure, the screen refreshes every three seconds.

C.1 Notes on Using the Integrated Console Facility

The line interface of the mainframe system console is not as fast as a normal 3270 full screen interface. As a result, if you want to use a mainframe console as a VM master console, you should pace the autologging of machines so that the console does not become overloaded with message traffic. In Figure 104 on page 154 is a sample REXX EXEC you can merge into your AUTOLOG1 PROFILE EXEC processing. If your system operator is logged on to the integrated console, and you are running first level, then a 4 second delay is added between each service machine autolog.

```
/* AUTOLOG1 PROFILE EXEC */
/* We'll avoid a flood of messages in on the Real System console*/
trace 'O'
address command
RunningFirstLevel= ( length(diag(0))=40  )
                   /* note: DIAG0 returns 40 bytes per VM level */
'CP XAUTOLOG user1'
call MaybeSleep
'CP XAUTOLOG user2'
call MaybeSleep
/*
Continue your XAUTOLOGing here
*/
exit

MAYBESLEEP:
 SysOper   = left(diag(8,'Q SYSOPER'),8)
 OnSYSCons = (left(word(diag(8,'Q USER' SysOper),3),4)='SYSC')
 If OnSysCons & RunningFirstLevel then 'CP SLEEP 4 SEC'
return
```

Figure 104. Sample REXX EXEC for use with Integrated Console Facility

The SCP command line can be a maximum of 126 characters. If your operators issue commands that exceed this limit, they will have to split these commands into command strings of fewer than 126 characters.

Appendix D. QUIESCE/RESUME Request with a 3990 Model 6

In Figure 105, you can see the logic flow used when a service representative (SR) initiates a quiesce/resume from the control panel on a 3990 Model 6. This example assumes no guest operating systems are operating at the time of the quiesce/resume request.

- SR makes quiesce request from control panel.
- Control unit produces asynchronous message for quiesce request.
- VM determines scope of request.
- VM checks that the requested channel paths are offline. If not, VM will vary the affected channel paths offline and mark all affected paths as quiesced.
- VM responds to control unit with "request completed successfully."
- VM issues message informing operator of quiesce.
- SR sees that request has been honored by all host systems.
- SR proceeds with service.
- SR makes resume request from control panel.
- Control unit produces asynchronous message for resume request.
- VM determines scope of request.
- VM varies online all paths that were marked quiesced by the quiesce request, un-marks these paths from being quiesced, and issues a message informing operator of resume.

Figure 105. 3990 Model 6 Quiesce/Resume Scenario

These are the new messages, which can appear at the operator console during a Quiesce/Resume request from the 3990 Model 6.

Message ID	Message text
HCPCUP6400	Quiesce request was successful for path (CHPID) to device(s) (RDEV RDEV...).
HCPCUP6401	Resume request was successful for path (CHPID) to device(s) (RDEV RDEV...).
HCPCUP6402(01)	Quiesce request failed because of a vary processing failure on path (CHPID) to device(s) (RDEV RDEV...).
HCPCUP6402(02)	Quiesce request failed because an error was detected in the request for path (CHPID) to device(s) (RDEV RDEV...).
HCPCUP6402(03)	Quiesce request failed because path (CHPID) is the last path to online device(s) (RDEV RDEV...).
HCPCUP6403(01)	Quiesce request failed because guest (USERID) did not respond for path (CHPID) to device (RDEV).
HCPCUP6403(02)	Quiesce request failed because guest (USERID) issued an unsuccessful response for path (CHPID) to device (RDEV).

HCPCUP6404(01) Resume request failed because of a vary processing failure on path (CHPID) to device(s) (RDEV RDEV...).

HCPCUP6404(02) Resume request failed because an error was detected on path (CHPID) to device(s) (RDEV RDEV...).

Appendix E. Monitor and Accounting Record Changes

This appendix provides details of monitor records and accounting record changes that have occurred as a result of enhancements made to VM/ESA Release 2.2. For more details on the enhancements themselves, please refer to the body of the document.

For further information on these and all the VM/ESA Monitor records, see the MONITOR LIST1403 file, which is located on MAINT's 194 minidisk.

E.1 Minidisk Caching Changes

These changes have been made to support the enhancements made to VM/ESA Release 2.2's full track minidisk caching. Please refer to section 2.1, "Minidisk Caching Enhancement" on page 17 for further details.

Table 11. Changed Monitor Records for Minidisk Caching

Domain	Record	Description
0	14	Some fields no longer relevant and are reserved
0	14	Some fields mapped to new control block fields
0	3, 14	Additional fields added for new information
1	15	Additional fields added for new information
4	1, 2, 3, 9	Additional fields added for new information
6	3	Additional fields added for new information

E.2 Scheduler Enhancement Changes

These changes have been made to support the enhancements made to VM/ESA Release 2.2's CP scheduler. Please refer to section 2.3, "Share Capping and Proportional Distribution" on page 41 for further details.

Table 12 (Page 1 of 2). Monitor Records that have Changed for Scheduler Enhancements

Domain	Record	Description
0	10	Scheduler activity, sample record
0	12	User wait statues, sample record
1	15	Logged on user, sample configuration record
2	4	Add user to dispatch list, event record
2	5	Drop user from dispatch list, event record

Table 12 (Page 2 of 2). Monitor Records that have Changed for Scheduler Enhancements

Domain	Record	Description
2	6	Add user to eligible list, event record
2	9	SET SHARE changes, event record
4	1	User logon data, event record
4	2	User logoff data, event record
4	3	User activity data, sample record
4	4	User interaction data, sample record
4	9	User activity data at transaction end, event record

E.3 ADMF Support Changes

These changes have been made to support the enhancements made to VM/ESA Release 2.2 for ADMF Support. Please refer to section 2.7, “Asynchronous Data Mover Support” on page 63 for further details.

Table 13. Monitor Records that have Changed for ADMF Support

Domain	Record	Description
1	4	Operating system and special processing capabilities

E.4 Accounting Record Type 4

Accounting record Type 4 has been changed to register the BYUSER ID used to Logon BY the user ID. For more detail of the enhancements themselves, please refer to section 2.6, “Logon BY” on page 58.

Table 14. Changed Accounting Record Type 4

Start	End	Contents
1	8	User ID specified on the command.
9	16	Reserved for IBM use.
17	28	Date and time of accounting (mmddyyhhmmss).
29	32	Terminal address.
33	40	Invalid password.
41	48	User ID that entered AUTOLOG, XAUTOLOG, APPCVM CONNECT or the **BYUSER ID** that entered LOGON.
49	51	Reserved for IBM use.
52	53	Current invalid password count in hexadecimal.
54	55	Accounting record limit in hexadecimal.
56	56	Blank.
57	70	Reserved for IBM use.
71	78	LUNAME or SNA terminal.
79	80	Accounting card identification (04).

Appendix F. VMLINK Setup, Exit Activation and Using Examples

This appendix provides details about the VMLINK setup, exit activation, and using examples. It is a continuation of the topic started in 3.1, “VMLINK” on page 71.

F.1 VMLINK Implementation

VMLINK is used to link, access, release, and detach minidisks and SFS directories where products or applications reside. VMLINK will also link respective prerequisite and co-requisite disks, and handle most common invocation exit requirements.

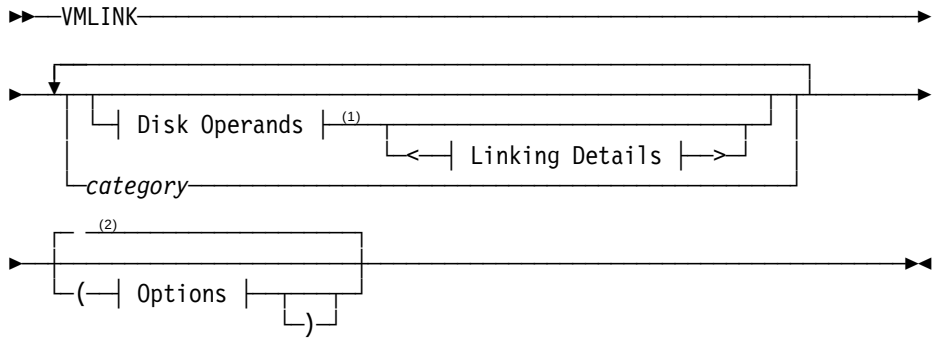
VMLINK not only has a user friendly panel interface, but it can also be issued as a simple command. Its command interface and internal design allow VMLINK to interact directly with REXX.

All the necessary information needed by VMLINK, like minidisks addresses or SFS directories, is stored in a NAMES file. Some VMLINK local default values, like the name of the VMLINK NAMES file, can be set on the VMLINK control file VMLINK CONTROL.

The implementation of VMLINK involved enhancements to the following CMS commands:

DEFAULTS Updated with the KEEP/NOKEEP, TYPE/NOTYPE, MODE0/NOMODE0, SAVE/NOSAVE options, and VMLINK profile defaults.
EXECUTE Updated so execute can process VMLINK panel commands.
NAMEFIND Was updated to look for the VMLINK names file constructs.
NAMES A new VMLINK option provides a panel interface for the VMLINK names files.

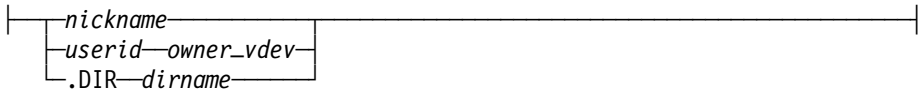
CMS VMLINK is a general user command and its syntax is shown in Figure 106.



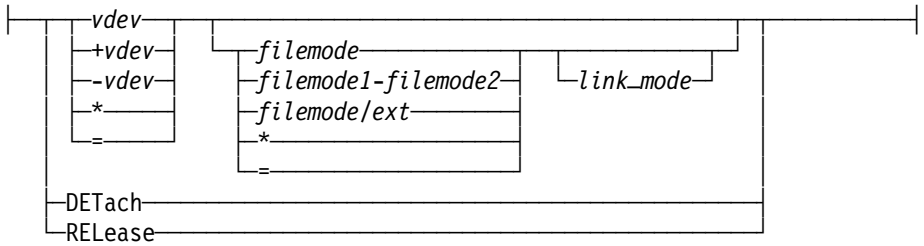
Notes:
1 If you do not specify any operands, the VMLINK menu is entered.
2 The default options are shown in the Options group.

Figure 106. CMS VMLINK Command Syntax

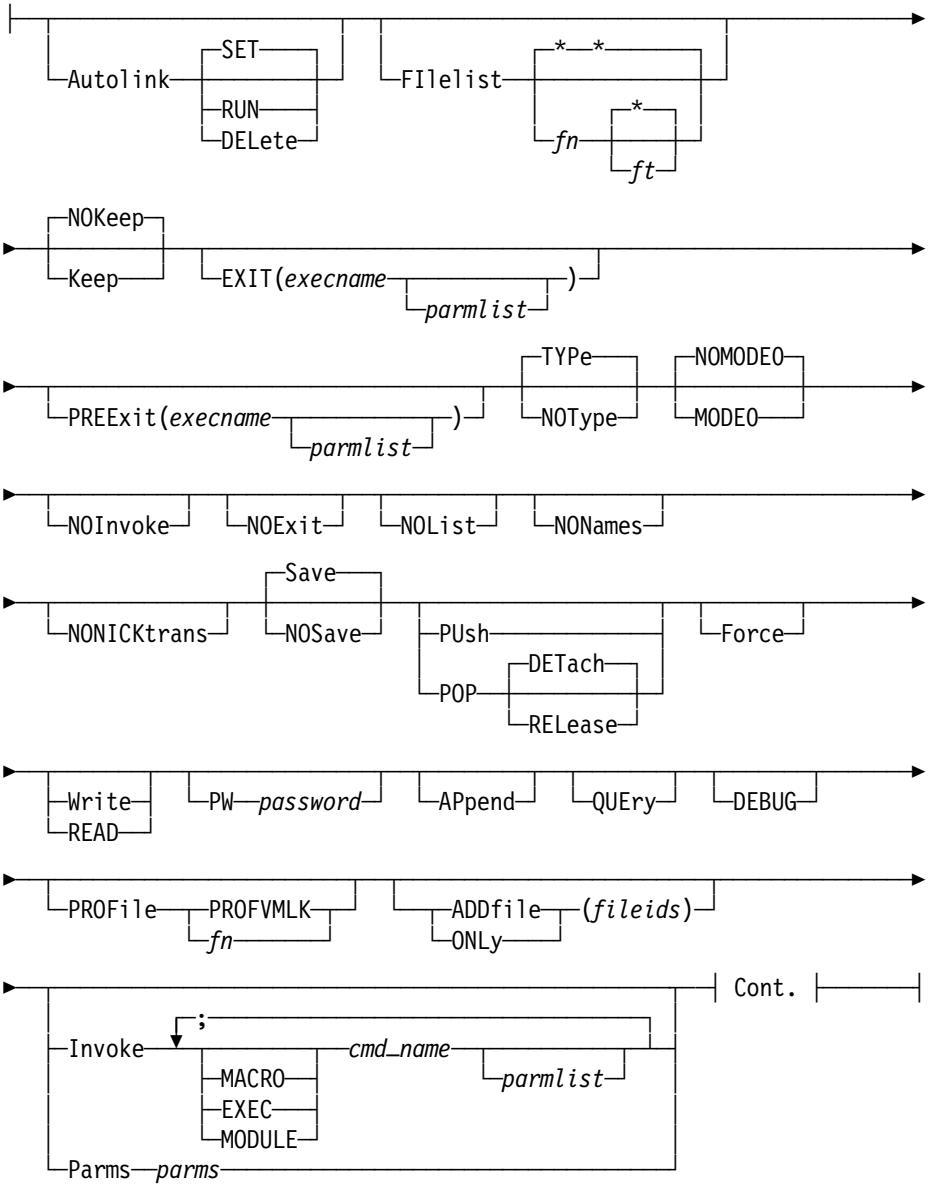
Disk Operands:



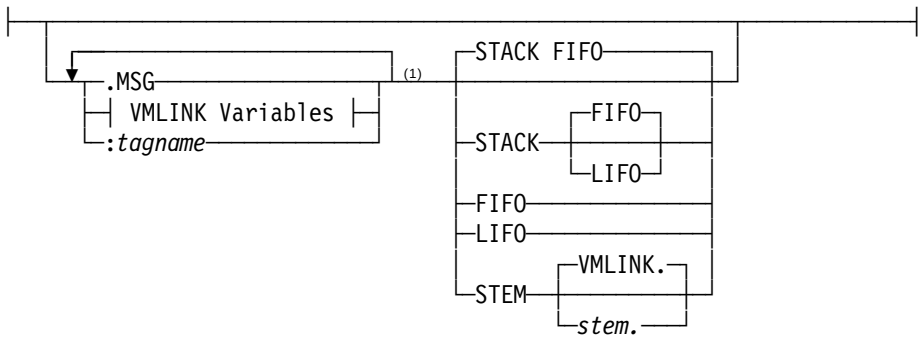
Linking Details:



Options:



Cont.:



VMLINK Variables:

Note:

[1] Messages, variables, and tag values are returned on the program stack or in REXX stem variables, as indicated by the STACK or STEM keywords.

Below are the operand descriptions for the VMLINK command syntax.

Disk Operand and Linking Details

nickname
is a nickname defined in a CMS NAMES file by a :NICK tag.

userid owner_vdev
identifies the user ID of the owner of a minidisk and the virtual device number as defined in the owner's user directory entry. When a minidisk is specified this way, use the NONAMES option so VMLINK does not search any NAMES files. Otherwise, VMLINK might match the user ID with a nickname.

.DIR *dirname*
identifies a directory.

vdev
is the virtual device number at which a minidisk is being linked. To select the next free virtual address in a range, specify the start of a range as *+vdev* or *-vdev* (ascending or descending order). An equals sign selects

the default virtual address. (See section F.1, “VMLINK Implementation” on page 159 and note 5 on page 167.) An asterisk selects the next free virtual address.

For SFS directories, the asterisk and equals sign are place holders.

filemode
is the filemode letter at which the minidisk or directory is being accessed. (See section F.1, “VMLINK Implementation” on page 159 and note 5 on page 167.) To have the first free file mode letter in a range assigned, specify the range as *filemode1-filemode2*. An asterisk selects the default range, Z-A, which means that the search starts at Z and progresses toward A. An equals sign selects the default file mode.

/ext
is the filemode of the parent minidisk or directory (read-only extension).

link_mode
specifies whether you get read-only or read/write access to the disk. For a minidisk, all modes defined for the CP LINK command are valid. The default is read-only (RR).

For SFS, you can specify FORCERO or FORCERW. If you do not own the directory, FORCERO is the default. If you own a file control directory, FORCERW is the default. If you own a directory control directory, FORCERW is the default unless someone else has the directory accessed R/W, in which case FORCERO is the default.

To have VMLINK use appropriate link modes for minidisks and directories associated with a nickname, use the READ and WRITE options. If a minidisk might already be linked R/W but not accessed, specify MR or use the WRITE option.

DETach
detaches a minidisk or releases a directory. DETACH is ignored if the POP option is used.

RELease
releases a minidisk or directory. RELEASE is ignored if the POP option is used.

category
is a category name. When a category name is specified, the VMLINK panel displays all nicknames in that category.

Command Options

Autolink/Autolink SET
puts the disk operand in a list of disks to be accessed when you logon or IPL CMS. The list is kept in the LASTING GLOBALV file. Options are not saved in the list, so nicknames must be in one of the default names.

This command also has the tendency to actually perform the link.

Autolink RUN
links the disks that were specified by either Autolink or the Autolink SET. Any operands or other options are ignored when this option is entered.

Autolink Delete
deletes a minidisk or directory from the list of disks to be accessed when you logon or IPL CMS. If you set the autolink with the WRITE option, use the WRITE option with the autolink DELETE so you do not delete a read-only entry in the list.

FIlelist *fn ft*
causes VMLINK to issue a FILELIST command for files on the first minidisk or SFS directory accessed. The default is FILELIST * * . It overrides any commands on the INVOKE tag. Use the KEEP option if you would like to keep the disks accessed after this command. If your other options follow this one on the VMLINK command, you must give a filename and filetype.

NOKeep
is used with INVOKE or FILELIST to restore disks to their previous state after VMLINK has finished. This is the default.

NOKEEP has no effect when either FILELIST or INVOKE option is specified.

Keep
is used with INVOKE or FILELIST to keep disks accessed after VMLINK has finished.

EXIT(*execname parms***)**
names an exit EXEC to be called after the minidisk or directory has been accessed.

Note: See the notes section for the PREEXIT option for additional information.

PREEXIT(*execname parms***)**
names an exit EXEC to be called following nickname resolution. The parameter list can include all the special variables except .LA.

Notes:

1. The exit EXEC is called separately for each disk accessed by the VMLINK command and has access to variable and tag data for that disk only.
2. A disk ID is always passed to the exit as the first argument.
3. A parameter list passed to an exit cannot include a right parentheses because the first right parenthesis encountered implies the end of the exit's parameter list. If a right parenthesis is needed, use the EXIT or PREEXIT tags in the VMLINK NAMES file rather than the option.
4. If a non-zero return code is returned from the exit EXEC, VMLINK ends. A return code of 12 terminates VMLINK with a return code of 0. Otherwise, VMLINK terminates with a return code in the form *3xxx*, where *xxx* is the return code from the exit.
5. This option overrides the equivalent tag in the NAMES file.
6. When the POP option with the DETACH or RELEASE operand is used, VMLINK does not call the exit EXECs.

Type
displays linking messages on the terminal. This is the default.

NOType
prevents linking messages from being displayed on the terminal.

NOMODE0
specifies that mode 0 links will not be done. This is the default.

MODE0
specifies that mode 0 links will be done if the ACCESSM0 command is available. This will allow you to view mode 0 files on the disk you access.

NOInvoke
overrides the INVOKE tag in the NAMES file.

NOExit
overrides the PREEXIT and EXIT tags in the NAMES file.

NOList
overrides the LIST tag in the NAMES file.

NONames
suppresses the search of the NAMES files. Use NONAMES when the disk is identified by name: *userid owner_vdev*.

NONICKtrans
tells VMLINK not to call the VMLNICXT EXEC. This is a locally customizable exit EXEC that replaces a specified nickname with another before the NAMES files are searched.

If VMLNICXT EXEC does not exist, it is not called.

Save
causes VMLINK to save the program stack. This is the default.

NOSave
prevents VMLINK from saving the program stack. Exits and invoked routines can then use the data on the stack.

PUsh
causes VMLINK to keep a record of its actions when it links and accesses minidisks and directories in read-only mode. A later call to VMLINK for a minidisk or directory with the POP option restores it to its former state.

POP/POP DETach
causes VMLINK to restore specified disks to their previous states, based on information saved by linking with the PUSH option. If no disk operand is specified with this option, all disks for which information has been saved are restored. POP is the same as POP DETach.

POP RELease
causes VMLINK to restore specified disks to their previous states, based on information saved by linking with the PUSH option, except that minidisks are not detached. If no disk operand is specified with this option, all disks for which information has been saved are restored.

Force
forces a link at *vdev*. If a device is attached at *vdev*, it is detached.

Write
accesses the disk in write mode. An existing read-only link to the same disk is not disturbed.

WRITE overrides *link_mode*.

READ
accesses the disk in read-only mode regardless of the NAMES file default. Minidisks are linked with the link mode of RR. Directories are accessed with FORCERO. For minidisks, an existing write link or access to the same disk is not disturbed provided a different virtual device number is supplied.

READ overrides *link_mode*.

PW *password*
specifies a password to be used for the CP LINK command. The password applies to all minidisks linked.

APpend
appends the list of nicknames to the list displayed on the VMLINK menu. This option can be used only within the menu.

QUEry
displays the NAMES file entries of the nicknames specified by the user, regardless of the nodes specified on the :NODE tag. When this option is used, the nickname is not processed; therefore, no accesses are done.

DEBUG
is provided to help aid in debugging of problems. It will help identify from which NAMES file the nickname entry was obtained. Also, any messages from CP or CMS are displayed.

PROFile *fn*
names the XEDIT profile for the VMLINK panel. The default is PROFVMLK. For more information on the PROFVMLK macro, see *VM/ESA CMS Command and Reference*.

ADDFile(*fileids***)**
appends additional files to the beginning of the search list of NAMES files. More than one file ID can be specified. The filetype and filemode of the last file name in the list can default to NAMES *. This option overrides the *ADDFILE control record.

ONLY(*fileids***)**
specifies which NAMES files should be searched. More than one file ID can be specified. The filetype and mode of the last filename in the list can default to NAMES *. This option overrides the *FILES control record.

Invoke *environment command parmlist*
names routines to be executed after all the disks have been accessed. The INVOKE option overrides the INVOKE tag in the NAMES files. It must be the last option specified. It is not executed if errors are detected in linking and accessing the disks.

Sets of *environment command parmlist* can be repeated, delimited by semicolons (;). The parameter list can include VMLINK variables. The environment is optional and defaults to CMS. It can be:

CMS to invoke *command* as though from the CMS command line. This is the default. CP commands must be preceded by “CP.”

MACRO to invoke *command* as an XEDIT macro.

EXEC to invoke *command* as an EXEC.

MODULE to invoke *command* in the command environment (like REXX/VM *ADDRESS COMMAND*). CP commands must be preceded by “CP,” and EXECs by “EXEC.”

If VMLINK receives a non-zero return code from a routine, VMLINK terminates with a return code in the form of *3xxx*, where *xxx* is the return code from the invoked routine.

PARMS *parms*
specifies parameters on the command invocation that are to be passed to exits and routines on the EXIT, PREEXIT, or INVOKE options or tag in the NAMES file. Use the .PA variable to specify where the parameters should be substituted. The parameter list that is passed can include all of the VMLINK variables.

All text following the PARMS option is considered part of the parameter list; PARMS must be the last option.

.MSG
puts VMLINK messages on the stack or in REXX stem variables. The default is to display messages at the terminal. Messages are returned before any other data.

tagname
returns the value assigned to the tag.

.AR
returns the arguments passed with VMLINK. The arguments include everything preceding the first left parenthesis.

.CA
returns the :CATEGORY tag.

.CU
returns the virtual device number used to link a minidisk or a fully qualified directory name (for example, DIR VMSYSU:MAINT.SAMPLES). At pre-exit time, a free virtual device number or directory name is available.

.FM
returns the filemode letter used to access the disk. At pre-exit time, a free filemode letter is available.

.IN
returns the :INVOKE tag value.

.LA
returns the minidisk label. At pre-exit time, the label is not available since the disk has not yet been accessed.

.LI
returns the :LIST tag value.

.NF
returns the file ID of the NAMES file in which VMLINK found the nickname.

.NI
returns the :NICK tag value.

.NO
returns the :NODE tag value.

.OP
returns the options passed with VMLINK. The options include everything following the first left parenthesis.

.PA
returns the text following the PARMS option keyword.

.PR
returns the :PRODUCT tag value.

.TA
returns the *tag* value pairs for tags specified in the options on the invocation of the VMLINK command.

.TI
returns the :TITLE value.

.XD
returns any data specified on the RETURN statement of an exit EXEC.

n
identifies a particular blank-delimited token in the requested string. For example, .OP3 refers to the third token in the options string. If *n* is not specified, the complete string is returned.

m
identifies the disk for which the information is requested. For example, .FM3 is the file mode letter of the third disk accessed by the VMLINK command. If *m* is not specified, 1 is assumed.

STACK FIFO/STACK/FIFO
specifies that the requested data should be stacked FIFO.

STACK LIFO/LIFO
specifies that the requested data should be stacked LIFO.

STEM *stem.*
specifies that the requested data should be returned to the calling program by setting a stem variable. If a stem name is not provided, VMLINK is used as the default.

F.1.1 Usage Notes

1. WRITE disk status cannot be saved or restored using the PUSH or POP options.
2. Link information is ignored when POP is specified.
3. If no disk operands are specified, a menu of all the nicknames valid on the system is displayed.
4. VMLINK determines how to process each disk operand, in turn, by trying to establish one of these conditions as true:
   a. The NONAMES option is specified. If so, the operands are assumed to be only minidisk IDs and directory names. No NAMES file is searched for nicknames.
   b. The operand is "DIR." If so, VMLINK processes the operand as part of a directory specification.
   c. The operand is a nickname that is valid on the node. If so, VMLINK processes it as described in section "NAMES Files Processing" on page 176.
   d. The operand matches the content of a CATEGORY tag that is valid for the node. If so, a menu of all the valid nicknames with a matching category is displayed.
   e. Otherwise, VMLINK tries to process the operand as part of a minidisk ID in the form, *userid vdev*.
5. The VMLINK control file, VMLINK CONTROL, sets local defaults for some VMLINK values. The defaults are overridden by values in a nickname entry, which in turn are overridden by operands specified with the command. (See section "VMLINK CONTROL File" on page 73).
6. VMLINK searches CMS NAMES files for nicknames. The search order and names of the NAMES files are defined in the VMLINK CONTROL control file. (See sections "VMLINK CONTROL File" on page 73 and "VMLINK NAMES Files" on page 172).

7. If you have called VMLINK without referring to a NAMES file, no exit EXEC is called.

F.1.2 VMLINK Control File and Names File

To customize VMLINK, it is advisable, but not required, to create one VMLINK CONTROL file to define the overall installation environment. This VMLINK CONTROL file, which will be written by the system support personnel, is called the *system control file*.

The system support personnel may need to create a VMLINK NAMES file for their installation. The VMLINK CONTROL file and the VMLINK NAMES file need to reside in a minidisk accessible to all the users that will use VMLINK.

One possible choice is the 19E Y-disk, though after every change to the VMLINK files, you will have to re-save CMS in order to keep the Y-STAT shared. But, frequent re-saving of CMS could cause many CMS NSS files with only a few users each. Another possibility, depending on the installation environment, is to choose another minidisk address, avoiding the need to re-save CMS. It is up to the system support personnel to decide where to implement the files, depending on their installations requirements.

The mentioned consideration also applies in the following cases:

- Installation customization of the PROFVMLK XEDIT macro, used to control the panel interface. It is not recommended that you update the PROFVMLK, but a new XEDIT macro should be modeled from PROFVMLK and the users defaults updated to use the replacement macro.
- Activation of the VMLINK exits.

VMLINK CONTROL Records

The VMLINK control file is a simple collection of control records grouped together in a flat file called VMLINK CONTROL.

Each control record begins with an asterisk and is followed by the control record name. This is the opposite of most conventions, where an asterisk (*) represents a comment. If you wish to comment out one of these records, double the asterisk (**). VMLINK ignores blank lines and lines that do not begin with a valid record name.

*ADDFILE Record

The *ADDFILE record lists NAMES files to be searched before the files listed on the *FILES record. In a local control file, a *FILES record would override the *FILES record in the system control file.

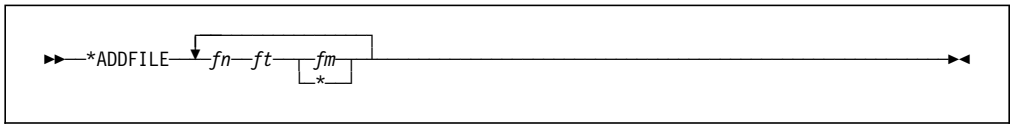


*Figure 107. VMLINK *ADDFILE Control Record*

*EQUATE Record

The *EQUATE record assigns a list of nodes to a symbol. When the symbol is used on the :NODE tag, the list of nodes is substituted for the symbol.

Notes:

1. *EQUATE records can be used to define more than one symbol, but VMLINK uses only the first instance of any particular value of *symbol*.
2. A *symbol* defined on an *EQUATE record cannot be used in the list of nodes on another *EQUATE record.

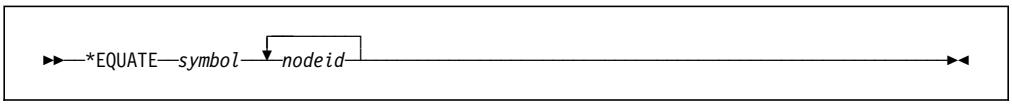


*Figure 108. VMLINK *EQUATE Control Record*

*ERROR Record

The *ERROR record determines whether VMLINK accesses any disks when it cannot access all the disks.

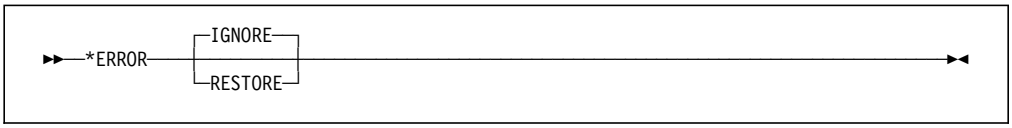


*Figure 109. VMLINK *ERROR Control Record*

where:

Option	Action
IGNORE	is the default and means to access as many disks as possible.
RESTORE	means to restore any disks accessed if a nonzero return code is received from LINK, ACCESS, or an exit routine.

*EXIT Record

The *EXIT record names an EXEC that is called after VMLINK has accessed the minidisk and SFS directories. The exit is called for each minidisk and directory accessed before the EXECs named on an EXIT option or tag.

The NOEXIT option does not override the *EXIT record.

Like the EXIT option and tag, the parameter list can include VMLINK variables.

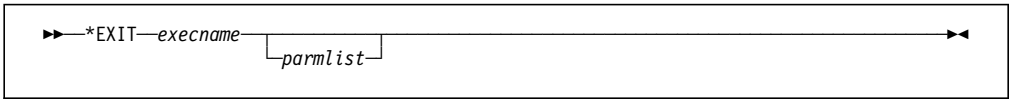


*Figure 110. VMLINK *EXIT Control Record*

*FILES Record

The *FILES record specifies the file ID and search order for the NAMES files.

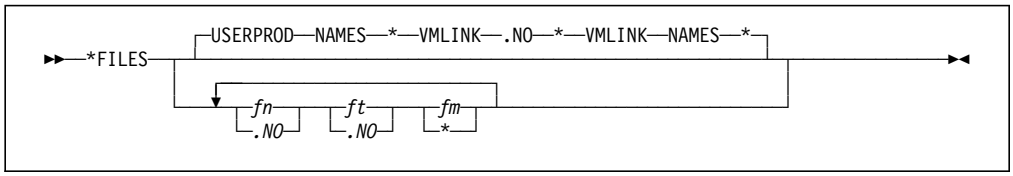


*Figure 111. VMLINK *FILES Control Record*

Note: VMLINK replaces the VMLINK variable .NO with the system node ID. (See section "*ID Record").

*ID Record

The *ID record determines how VMLINK obtains the system node ID.

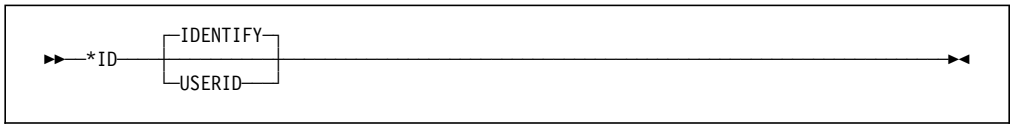


*Figure 112. VMLINK *ID Control Record*

where:

Option	Action
IDENTIFY	is the default and means to use the CMS IDENTIFY command.
USERID	means to use the CP QUERY USERID command.

*MODES Record

The *MODES record defines the default ranges and search orders for finding a free file mode letter. The default search is used when an asterisk is specified for the file mode in the linking details.

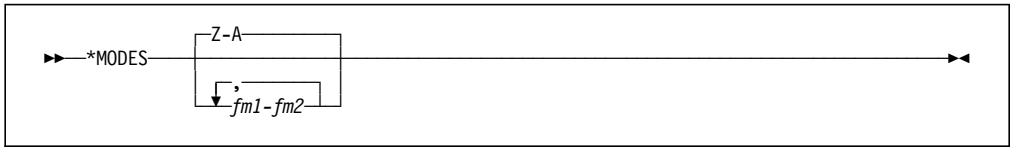


*Figure 113. VMLINK *MODES Control Record*

*PEXIT Record

The *PEXIT record specifies a pre-exit EXEC that is called after VMLINK has located a free virtual device number and file mode letter and before the disk is accessed. This exit is called for each minidisk and directory accessed before the EXECs named on the PREEXIT option and tag. The NOEXIT option does not override the *PEXIT record.

Like the PREEXIT option and tag, the parameter list can include VMLINK variables.

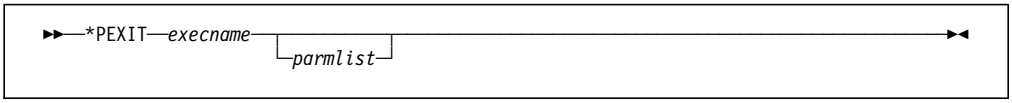


*Figure 114. VMLINK *PEXIT Control Record*

*VDEV Record

The *VDEV record defines where to start searching for a free virtual device number. The default range is used when an asterisk is specified for the virtual device number in the linking details. As with normal virtual device numbers, it is hexadecimal.

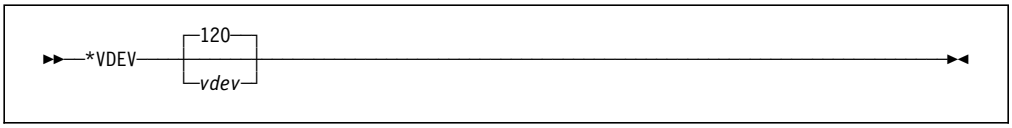


*Figure 115. VMLINK *VDEV Control Record*

VMLINK CONTROL File Example

```
************************************************************
** Name     : VMLINK   CONTROL                             **
**                                                         **
** Function : The definitions in this file will define the **
**            system defaults.  If this file does not exist**
**            or a record is not specified, a default value**
**            will be used.                                **
**                                                         **
**  For performance reasons, it is recommended that only   **
**  values which are to be changed are included in this    **
**  file.  Also, comment records can be removed.           **
**                                                         **
**  Each record of this file is assumed to be a comment,   **
**  unless it starts with a valid keyword.                 **
**  Valid KEYWORDS are:                                    **
**                                                         **
**  ADDFILE - EQUATE - ERROR - EXIT - FILES - ID - MODES   **
**                  PEXIT  -  VDEV                         **
**                                                         **
**  All of them preceded by an '*'                         **
************************************************************
*EQUATE VMXA  ROMEPPC RMXARNSL
*EQUATE ESA11 RNSLESA RMESNPM2 RMESNPM4
*EQUATE ESA21 RMESNPM5 RMESA121

*ERROR RESTORE

*EXIT EXITCTL
*PEXIT PEXITCTL

*ADDFILE NEWFILE
*FILES USERPROD NAMES * MYOWN NAMES * PRODUCT .NO * VMLINK NAMES S

** For discussion purposes, we show a default value on *ID
*ID IDENTIFY

*MODES W-Z
*VDEV 900
```

Figure 116. VMLINK CONTROL Example File

In the VMLINK CONTROL file example shown in Figure 116, some defaults were set:

- The **VMXA** symbol was defined, through the *EQUATE control record, to be a synonym of ROMEPPC and RMXARNSL nodes. In the same way, **ESA11** was defined for RNSLESA, RMESNPM2 and RMESNPM4, and **ESA21** for RMSNPM5 and RMESA121.

  The list of nodes assigned to a *symbol* might have significance when systems can be grouped by some common characteristics, for example, system level maintenance, system level release, products activated, and so on.

- The **RESTORE** option on the *ERROR control record tells VMLINK not to link as many minidisks as possible, but to restore any minidisk accessed if a nonzero return code is received.

- The **EXITCTL** exit defined on the *EXIT control record is the VMLINK exit EXEC that will be executed after a minidisk is linked and accessed.

- If there is any **NEWFILE NAMES** file found when invoking VMLINK, it will be searched before the files listed on the *FILES control record.

- On the *FILES control record, many VMLINK NAMES files were specific, but their order determines the search order. In the example, the first NAMES file to be searched is the default one, USERPROD NAMES, then MYOWN NAMES file, and so on.

  Refer to section “VMLINK NAMES File Examples” on page 177 for information about the creation of **MYOWN NAMES** file.

- The IDENTIFY option of the *ID control record means that VMLINK will use the CMS IDENTIFY command to obtain the system node ID.

- The way our *MODES control record is defined in the example, the first default filemode that will be used to access a minidisk is **W**, but if W is unavailable, then the search for a free filemode will continue incrementally towards **Z**, ultimately ending in error message DMSVML1277E (no filemode is available) when you run out of access filemodes.

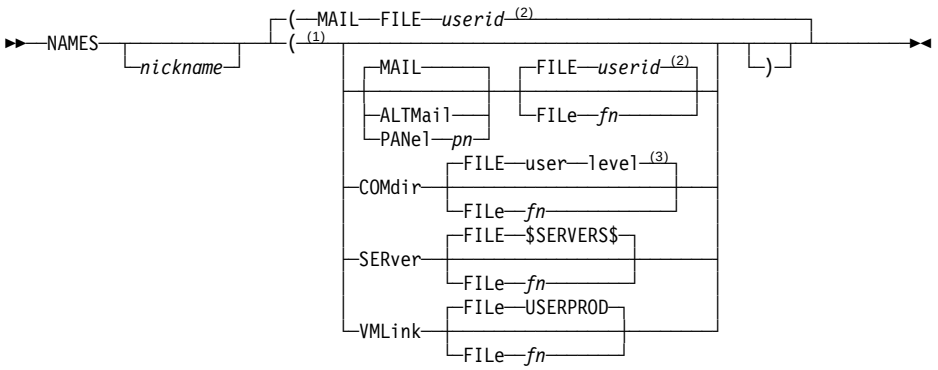
- The **PEXITCTL** exit defined on the *PEXIT control record is the VMLINK preexit EXEC that will be executed before a minidisk is linked and accessed.

- The defaults virtual addresses for linking minidisks are defined on the *VDEV control record. The 900 virtual address was chosen in the example.

VMLINK NAMES Files

The default search order and file IDs of the NAMES files are defined in the *FILES record in the VMLINK CONTROL file (see Figure 111 on page 170). The default list can be changed by the ONLY and ADDFILE options of the VMLINK command. Unless the disk operands are minidisk IDs or directories, VMLINK requires that at least one of the NAMES files in the list be accessed.

As seen in Figure 117 on page 173, a new VMLINK option was added to the NAMES command. See *VM/ESA CMS command Reference* for details of the command changes.



Notes:

1. You can enter Options in any order between the parentheses.
2. The parameter “userid” is your virtual machine user ID.
3. Default is the file name of the “user-level” communications directory file, if any; otherwise, it is “UCOMDIR.”

Figure 117. VMLINK NAMES Command Enhancement

Also, a new panel interface was added in VM/ESA 2.2 for the creation and management of the VMLINK names files.

The following is an example of the simple CMS NAMES Command:

NAMES

```
 ====> NAMES (Mail panel)      File: VERONICA NAMES    A0               <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: Martin                Notebook: Martin
Userid: STROHMAI
  Node: WTSCPOK
                      Name: Martin Strohmaier
                     Phone: (914) 433 1517
                   Address: Poughkeepsie
                          :
                          :
                          :
             List of Names:
                          :
```

Figure 118. CMS NAMES Panel Interface

The following is an example of the CMS NAMES command with the VMLINK option:

```
NAMES    ( VMLINK
```

```
 ====> NAMES (Vmlink panel)    File: USERPROD NAMES    A0         <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname:          Description:
Product Linking    :
Information (USERID:
cuu/.DIR dirname)  :
Category:
                  Invoke:
                 Preexit:
                    Exit:
              Valid Nodes:
                        :
             List of Names:
                        :
                        :
                        :
 Tag:               Value:
 Tag:               Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                      ======> Screen 1 of 1 <======
```

Figure 119. VMLINK NAMES Panel Interface

As can be seen in Figure 119, the NAMES command with the VMLINK option invokes a tailored panel for creating the VMLINK NAMES default file, USERPROD NAMES. The information collected in the VMLINK names file provides minidisk linking and SFS directory accessing with a nickname.

One nickname entry consists of at least a NICK tag and a PRODUCT tag or a NICK tag and a LIST tag. Other tags are optional. If a tag is missing, its default value is null.

Other tags located in the VMLINK NAMES file are:

Tag	Description
category:	*name* Associates this disk with other disks. If there is no *category* tag or if *name* begins with an at-sign (@) or percent-sign (%), the nickname entry is not displayed on the menu. VMLINK uses only the first eight characters of the category name. There can be more than one *category* tag in a nickname entry.
exit:	*execname parmlist* Calls exit EXECs after the disk is accessed and before the INVOKE option or tag is processed. The content and behavior of the tag is the same as the EXIT option on the VMLINK command. The NOEXIT and EXIT options override this tag.

invoke: *envir cmd parmlist;...*

Names routines to be invoked after the disk is accessed. The content and behavior of the tag is the same as the INVOKE option. Use the NOINVOKE or INVOKE option of the VMLINK command to override the tag.

Only the INVOKE tag assigned to the first nickname specified on the command is processed. INVOKE tags for nicknames on LIST tags are not processed.

The INVOKE tag is not executed if a non-zero return code is encountered during the process of accessing the disks.

list: *disk_operands*

Names additional disks to be accessed with this nickname. The disk operands are the same as for the VMLINK command, including nicknames. Shown as List of Names: on the VMLINK names panel interface.

The NOLIST option of the VMLINK command overrides this tag.

nick: Defines the disk's nickname. Shown as Nickname: on the VMLINK names panel interface.

node: *node_list*

Lists the nodes on which the nickname is valid. If there is no NODE: tag, the nickname is valid on all systems. The node list can contain node IDs, names defined on an *EQUATE record in the VMLINK CONTROL file, and node IDs qualified by an asterisk or a logical NOT sign:

- The asterisk matches any string at the beginning or end of a node ID. For example, **node:**ROM* matches any node ID starting with "ROM," and **node:*** matches any node ID.
- The logical NOT sign before a node ID specifically excludes that node. For example, **node:**¬ ROMELAB. A minus (-) sign works the same way as the not sign.
- The asterisk and the ¬ (logical NOT sign) cannot be combined in one node ID. A minus (-) sign works the same way as the not sign.

  Shown as Valid Nodes: on the VMLINK names panel interface.

preexit: *execname parmlist*

Calls exit EXECs before a minidisk is linked or a directory is accessed. Its behavior is the same as the PREEXIT option.

The NOEXIT and EXIT options override this tag.

product: *disk_operands linking details*

Is the linking information for the minidisk or SFS directory. The disk operands and linking details are the same as for the VMLINK command, except that the disk operand cannot be a nickname.

Shown as the three Product Linking: fields on the VMLINK names panel interface. All three fields after Nickname: and before Category: are treated as one field.

title: *description*

Describes the disk. The title is displayed on the terminal when the disk is accessed, and on the menu. When there is no TITLE tag, the nickname is not displayed on the menu. For the terminal, VMLINK generates a title from the disk or directory specification.

Shown as the Description: field on the VMLINK names panel interface.

tagname: *string*

Is a user tag. User tags must follow the VMLINK tags in the NAMES file. VMLINK takes no action based on the presence of a user tag, but the information it contains is available to VMLINK, exit, EXECs, and invoked routines.

Treated as a composite of the Tag: and Value: fields on the VMLINK names panel interface.

NAMES Files Processing

As shown in section "VMLINK Defaults Override" on page 74, VMLINK has its own defaults override hierarchy. For the NAMES file processing, VMLINK takes the following steps:

1. VMLINK searches for nicknames in any NAMES files specified on the command with the ADDFILE option.
2. VMLINK searches any files specified on the *ADDFILE record in VMLINK CONTROL.
3. VMLINK searches any files specified on the *FILE record in VMLINK CONTROL.

Files are searched in the order specified, and when a mode letter is not specified, they are searched in A-to-Z file mode order. Alternatively, you can limit the search to files specified with the ONLY option.

The search for a nickname entry ends only when a nickname entry is found that is valid for the node, or when all the NAMES files have been searched. If no match is found for a nickname entered on the command, VMLINK looks for matching *category* tags. VMLINK displays a menu of nicknames in the *category* before it processes the next nickname.

When a nickname is found in a NAMES file:

1. VMLINK verifies that the entry applies to the current node. If the filename of the NAMES file is the node ID, the nickname is valid. With other NAMES files, VMLINK verifies that the current node is included in the node IDs listed on the NODE tag. If the nickname does not apply to your system, VMLINK continues the search.
2. The contents of the LIST tag are appended to the list of disks to be accessed.
3. The EXECs listed on the PREEXIT tag are executed.
4. The minidisk or directory on the PRODUCT tag is linked and accessed.

5. The EXECs listed on the EXIT tag are executed.
6. VMLINK repeats the process to this point for the next nickname in the list.
7. When all nicknames have been processed, the INVOKE option or the INVOKE tag on the first nickname in the command is processed.

VMLINK NAMES File Examples

Though the default NAMES file for VMLINK is USERPROD NAMES, for the examples, MYOWN NAMES was used to show that it is possible to choose a unique name for the VMLINK NAMES file.

Notes:

1. If you want to create another file rather than the default USERPROD NAMES, you should issue the NAMES (VMLINK FILE MYOWN command. But in this case, it is important to remember that the file MYOWN NAMES *fm* should be declared on the *ADDFILE or *FILES control record of the VMLINK CONTROL file. See sections "*ADDFILE Record" on page 168 and "*FILES Record" on page 170 for details.
2. To enter the NAMES panel for filling in the VMLINK information and use the MYOWN NAMES file, the following command was issued:

   ```
   NAMES ( VMLINK FILE MYOWN
   ```

   where:

   VMLINK Is the NAMES command option for creating VMLINK entries.

   MYOWN Is the filename of the NAMES file for VMLINK. (See Figure 116 on page 171 for VMLINK CONTROL consistency.)

Understanding the following examples, which form the basis for the remaining VMLINK examples in this book, will aid you in your understanding of the VMLINK command.

The following examples show extreme and common situations where:

- Products or applications usually reside on more than one minidisk.
- Products or applications have other products or applications as prerequisites.

The examples use generic names for the program products and disks. The intent was to make the examples as general as possible. However, some assumptions were needed to make the examples as real as possible.

While introducing the examples, panels and assumptions will be explained.

Example 1:

This example represents a method to create a names entry for PRODUCT A. The example shows the names panel. The highlighted values are the user-entered fields.

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODA    Description: Linking PRODUCT A with its pre-req ...
Product Linking    :
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke:
                 Preexit:
                    Exit:
             Valid Nodes: TOTVM1 RMESNPM1
                        :
           List of Names: PRODA191 PRODC193
                        :
                        :
                        :
 Tag:              Value:
 Tag:              Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 120. VMLINK PRODUCT A NAMES File Entry

In Figure 120, PRODUCT A was defined with the following characteristics:

- The Nickname **PRODA** was chosen to define the PRODUCT A and its characteristics.
- The Description: field contains a description of the PRODUCT A product.
- A Category of **PRODUCTS** was assigned to PRODUCT A.
- The valid nodes where the nickname PRODA is valid are: **TOTVM1** and **RMESNPM1**.
- PRODUCT A has only one minidisk, and its entry is declared on the list of names as **PRODA191**. The PRODUCT A 191 minidisk is defined in another NAMES file entry in Figure 121 on page 179.
- PRODUCT A has as a link to the PRODUCT C 193 minidisk. This requirement is also declared on the list of names as **PRODC193**. The PRODC 193 minidisk is defined in another MYOWN NAMES file entry in Figure 128 on page 184. For the VMLINK execution of this link, see Figure 139 on page 197.

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODA191 Description: Linking Product A Mdisk 191 ...
Product Linking    : PRODUCTA 191 <+800 G-A MR>
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke:
                 Preexit:
                    Exit:
              Valid Nodes: TOTVM1 RMESNPM1
                         :
            List of Names:
                         :
                         :
                         :
 Tag:               Value:
 Tag:               Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 121. VMLINK PRODUCT A Minidisk 191 NAMES File Entry

The meaning of the new fields filled in Figure 121 follows:

- The nickname PRODA191 was chosen to match one of the entries on the List of Names in Figure 120 on page 178.
- The Product Linking: field is filled in.

  This is the most important difference between the declaration of the product, PRODUCT A, and the declaration of one of its minidisks.

  In Figure 120 on page 178, the *Product Linking* entry was not filled in because it was only a declaration of the product. This is the way to manage multiple disks required by a product. For that reason, on the *List of Names*, two entries were added: PRODA191 and PRODC193.

  Since the PRODA191 nickname entry is a minidisk declaration, the information about what and how to get the minidisk is needed. For this reason, on the Product Linking entry for PRODA191 in Figure 121 it stated:

  PRODUCTA 191 <+800 G-A MR>

  where:

  Operand Description

  PRODUCTA is the name of the virtual machine where the product resides.

  191 is the virtual address of the virtual machine PRODUCTA whose link is required.

  < is a required starting delimiter that must precede the virtual address operand.

  +800 is the target address range that the minidisk will be linked as. The +800 means to select the next free address starting at 800 and, if it is used, to continue the search for a free address, in ascending order.

  G-A is the filemode letter at which the minidisk being accessed. The search for a free filemode letter will begin at G and, if its busy, the search for a free filemode letter will continue towards A.

MR is the link mode for the minidisk. The default is RR, or read only.

> is a required ending delimiter that must follow the link mode operand.

Example 2:

This example presents the entries for PRODUCT B:

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6               <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODB    Description: Linking PRODUCT B with no pre-req ...
Product Linking    :
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke: EXEC INVOKEB .no1
                 Preexit: PREXITB .CA1
                    Exit:
              Valid Nodes: TOTVM1 RMESNPM6
                         :
             List of Names: PRODB191 PRODB193 PRODB195
                         :
                         :
                         :
 Tag:               Value:
 Tag:               Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 122. VMLINK PRODUCT B NAMES File Entry

The meaning of the new fields in Figure 122 as compared to those in Figure 120 on page 178 follows:

- Usually, it is useful or even mandatory, to get some task done just after all the necessary minidisks are linked and accessed. An example of this situation is the ISPF with its ISPSTART or ICU with its ADMCHART.

  VMLINK, through the INVOKE facility, allows you to transparently execute a “bootstrap” program. In Figure 122, an EXEC was choosen as the PRODUCT B bootstrap, INVOKEB. VMLINK also passed INVOKEB one parameter, **.no1**, which is a VMLINK variable representing the NODE tag value.

  The first parameter, **EXEC**, was required since INVOKEB is an EXEC. VMLINK INVOKE facility allows CMS and CP commands, macros, EXECs, and modules to be executed.

  To see how this VMLINK INVOKE facility works, VMLINK variables and the EXEC receiving the parameters, please refer to Figure 137 on page 193 and Figure 138 on page 195.

- Sometimes it is useful to obtain VMLINK information (through the VMLINK variables) or to execute some tasks after the nickname resolution and before the minidisks are linked and accessed.

  VMLINK also offers this facility through its *PREEXIT* exit. The PREEXIT for PRODUCT B will be an EXEC called **PEXITB**, which will receive the **.CA1** VMLINK variable as a parameter.

  To see how this VMLINK PREEXIT works, and see the VMLINK variables and the EXEC receiving the parameters, please refer to Figure 137 on page 193 and Figure 138 on page 195.

- PRODUCT B has three minidisks and their entries are declared on the List of Names: field as PRODB191 PRODB193 and PRODB195. Each of these minidisks is defined in Figure 123, Figure 124, and Figure 125, respectively.

  It is important to note that the List of Names: tag can be used either to define a product's minidisks (see Figure 122 on page 180) or to define a product's requisites (see Figure 120 on page 178).

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODB191 Description: Linking Product B Mdisk 191 ...
Product Linking    : PRODUCTB 191 <+500 E-L RR>
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke:
                 Preexit:
                    Exit: EXITB .CU1 .FM1
             Valid Nodes: TOTVM1 RMESNPM6
                        :
           List of Names:
                        :
                        :
                        :
 Tag:              Value:
 Tag:              Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 123. VMLINK PRODUCT B Minidisk 191 NAMES File Entry

The meaning of the new fields used in Figure 123 follows:

- VMLINK gives the opportunity to do some tasks after the minidisks are linked and accessed. This is implemented through the VMLINK exits.

  In Figure 123, the exit EXEC is **EXITB**. It will receive, as parameters, the VMLINK variables **.CU1** (virtual address used to link the minidisk) and **.FM1** (filemode letter used to access the minidisk).

  The main difference between the VMLINK EXIT and the INVOKE facility is that the EXIT only executes an EXEC, while INVOKE (through its environment parameter) is able to execute CMS and CP commands and macros, EXECs and modules.

  To see how this VMLINK exit works, the VMLINK variables and the EXEC receiving the parameters, please refer to Figure 137 and Figure 138 on page 195.

In Figure 124 and Figure 125, the fields for defining the remaining PRODUCT B minidisks are shown filled in.

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODB193 Description: Linking Product B Mdisk 193 ...
Product Linking    : PRODUCTB 193 <+500 E-L RR>
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke:
                 Preexit:
                    Exit: EXITB .CU1 .FM1
              Valid Nodes: TOTVM1 RMESNPM6
                         :
            List of Names:
                         :
                         :
                         :
 Tag:              Value:
 Tag:              Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 124. VMLINK PRODUCT B Minidisk 193 NAMES File Entry

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODB195 Description: Linking Product B Mdisk 195 ...
Product Linking    : PRODUCTB 195 <+500 E-L RR>
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke:
                 Preexit:
                    Exit: EXITB .CU1 .FM1
              Valid Nodes: TOTVM1 RMESNPM6
                         :
            List of Names:
                         :
                         :
                         :
 Tag:              Value:
 Tag:              Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 125. VMLINK PRODUCT B Minidisk 195 NAMES File Entry

Example 3:

This example presents the entries for PRODUCT C.

No new fields were filled in Figure 126, Figure 127 on page 183 and Figure 128 on page 184. They are provided for reference, as their contents are used throughout this section.

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODC    Description: Linking PRODUCT C with no pre-req ...
Product Linking    :
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke:
                 Preexit:
                    Exit:
              Valid Nodes: TOTVM1 RMESNPM1
                         :
             List of Names: PRODC191 PRODC193
                         :
                         :
                         :
 Tag:               Value:
 Tag:               Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 126. VMLINK PRODUCT C NAMES File Entry

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODC191 Description: Linking Product C Mdisk 191 ...
Product Linking    : PRODUCTC 191 <-200 X-Z RR>
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke:
                 Preexit:
                    Exit:
              Valid Nodes: TOTVM1 RMESNPM1
                         :
             List of Names:
                         :
                         :
                         :
 Tag:               Value:
 Tag:               Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 127. VMLINK PRODUCT C Minidisk 191 NAMES File Entry

```
 ====> NAMES (Vmlink panel)    File: MYOWN    NAMES    A6              <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODC193Description: Linking Product C Mdisk 193 ...
Product Linking    : PRODUCTC 193 <-200 X-Z RR>
Information (USERID:
cuu/.DIR dirname)  :
Category: PRODUCTS
                  Invoke:
                 Preexit:
                    Exit:
             Valid Nodes: TOTVM1 RMESNPM1
                        :
           List of Names:
                        :
                        :
                        :
 Tag:              Value:
 Tag:              Value:

1= Help       2= Add      3= Quit       4= Clear       5= Find      6= Change
7= PrevNick   8= NextNick 9= Delete    10= PrevScrn   11= NextScrn 12= Cursor
                                       ======> Screen 1 of 1 <======
```

Figure 128. VMLINK PRODUCT C Minidisk 193 NAMES File Entry

PROFVMLK XEDIT

When the VMLINK command is entered without operands, the VMLINK panel interface is entered. From the VMLINK panel interface, some of the possibilities available are:

- Link a product.
- Detach a product.
- List a product's category.
- Sort by different fields, such as category, name, description, and so on, by just pressing a PF key.

VMLINK executes the PROFVMLK XEDIT macro. But if you should like to change the PF Keys, or even the panel interface, you can create your own macro and invoke it with the PROFILE option of the VMLINK command, or use the CMS DEFAULTS command to change the default macro to one of your design.

For more information on the **PROFVMLK** macro, refer to *VM/ESA CMS Command Reference*.

VMLINK Exits

VMLINK provides three different exits that are activated at different stages of the VMLINK cycle. System support personnel need to code and activate them as their installation requires.

PREEXIT: This exit is activated after the VMLINK nickname resolution and before the minidisk linking and accessing. This pre-EXIT can be specified on the *PEXIT control record of the VMLINK Control File (see section "*PEXIT Record" on page 170), or in the PREEXIT: tag of the NAMES panel (see section "VMLINK NAMES Files" on page 172), or by issuing the VMLINK command with the PREEXIT option (see section F.1, "VMLINK Implementation" on page 159).

The PREEXIT exit will call an EXEC. Coding examples are given in Figure 132 and Figure 135.

EXIT: This exit is activated after the VMLINK has accessed the minidisks or SFS directories. This EXIT can be specified on the *EXIT control record of the VMLINK Control File (see section "*EXIT Record" on page 169), or in the EXIT: tag of the NAMES panel (see section "VMLINK NAMES Files" on page 172), or just by issuing the VMLINK command with the EXIT option (see section F.1, "VMLINK Implementation" on page 159).

The EXIT will call an EXEC. Coding examples are given in Figure 133 and Figure 136.

INVOKE: This exit is activated after the VMLINK has accessed the minidisks or SFS directories. Use the INVOKE: tag of the NAMES panel (see section "VMLINK NAMES Files" on page 172), or the VMLINK command with the INVOKE option (see section F.1, "VMLINK Implementation" on page 159) to specify what to invoke.

The main difference between the EXIT and the INVOKE exits is the EXIT only executes EXECs, while the INVOKE exit can execute CMS and CP commands, EXECs, macros, and modules.

An example is given in Figure 134 on page 191.

Table 15. VMLINK Exits Table References.

For details on:	Refer to:
VMLINK NAMES file	Section "VMLINK NAMES Files" on page 172
Exits examples	Figure 132 on page 191, Figure 133 on page 191, Figure 134 on page 191, Figure 135 on page 192 and Figure 136 on page 192.
Defaults override	Section "VMLINK Defaults Override" on page 74
Defaults override examples	Figure 138 on page 195
VMLINK exits	*VM/ESA CMS Command Reference*

F.2 Using VMLINK

Once you have completed any customization needed for your system, you are ready to start using VMLINK.

There are different ways to use VMLINK:

- Through the panel interface
- Using the CMS VMLINK command
- With autolink.

In the first two cases, the panel interface and CMS command, care should be taken to understand the hierarchy of the default overrides between VMLINK CONTROL, VMLINK NAMES files, and VMLINK command.

For clarity, the execution of the default override hierarchy will be shown in the VMLINK commands in section F.2.2, "Using VMLINK with Command Options" on page 190.

F.2.1 Using VMLINK with the Panel Interface

When issuing the CMS command VMLINK with no operands, VMLINK displays a list of nicknames in a panel interface, with a wide set of PF keys to perform many different tasks. This is much like the way FILELIST creates a list of your files, and has function specific PF keys. This panel interface is created by the execution of the PROFVMLK XEDIT macro. Refer to section “PROFVMLK XEDIT” on page 184 for more details about the PROFVMLK XEDIT macro.

The full power of XEDIT is available to the user while issuing commands against the list of nicknames displayed in the panel interface. However, some XEDIT subcommands are inappropriate in this environment. Subcommands that alter the format or the contents of a panel might cause unpredictable results. Some cases of these subcommands are: SET TRUNC, SET FNAME, SET FTYPE, SET FMODE, SET LINEND, and others.

- **Saving a List of Files:**

  Just as with FILELIST, VMLINK′s panel is an XEDIT file, so you can save it on a disk with the XEDIT SAVE or FILE commands. A file created this way will have the user ID as the default filename and VMLINK as the default filetype.

- **Issuing a Command From the List:**

  You can execute CMS commands through the VMLINK panel interface on the command line, but to prevent XEDIT from decoding the command, prefix the command with *CMS*.

  On the VMLINK panel, commands can be issued directly from the line on which a nickname is displayed by typing under the CMD column.

  At times, it may be necessary to execute commands from the command line before the commands typed against the list. An example can be a CMS QUERY DISK command, to look at what is accessed before linking a new product.

  To do this, PF12 can be used to move the cursor to the command line (instead of the Enter key). Then the command line command can be typed and executed by pressing the Enter key. The command entered from the command line will be executed while the command entered beside a nickname will not. PF12 can be used to move the cursor back to its previous position.

- **Using Symbols as Part of a Command**

  Symbols can be used to represent operands in a panel interface command. They are entered with the commands, directly on the screen. Symbols are needed if the command to be executed has operands or options that follow the nickname.

  The following symbols can be used:

  Symbol Description

  / means the nickname, <vdev, mode and link mode>.

  Note: The **< >** shown above is part of what is substituted in the command being issued.

  /n means the nickname.

  /v means the virtual device address.

  /m means the file mode.

/l means the link mode.

/o will execute the line as is and omit appending anything.

Any combinations of symbols can be used. For example:

Symbols Description

/n /v means the nickname followed by the virtual address.

/nv means the nickname followed by the virtual address.

/nvm means the nickname, virtual address and file mode.

If the symbol '/' appears in a command or in its operands, it must be issued from the command line.

VMLINK Panel Interface Example

The following example shows what a VMLINK panel might look like when system support has many products installed. To get this panel, the VMLINK command was entered with no options.

```
 VERONICA VMLINK   AO  V 260  Trunc=260 Size=188 Line=56 Col=1 Alt=0
Cmd   Nickname Vdev Fm Ext Lm  Category  Description
      CDSDSK   +200 * /        ALL       CDS Shadow disk.
      CFSEARCH +200 P /        ALL       Contextual File Search
      CLIB200  =    = /        OFFICE    ECFORMS Processing Code Disk
      CLIB300  =    = /        OFFICE    ECFORMS Forms Design Code Disk
      CLIB400  =    = /        OFFICE    ECFORMS Forms Admin. Code Disk
      COBOL2   =    = /        ALL       Cobol II Version 1 Release 3.2
      COLIS    +300 * /        ALL       Colis, PHONE, CALLUP disk.
      CORPDIR  =    = /        OFFICE    Corporate Directives
      CSP191   +291 E / A      ALL       A-disk
      CSP193   +293 B /        ALL       B-disk
      CSP195   +295 M /        ALL       M-disk
      CSP502   +202 C /        ALL       VSAM C-disk
      CSP503   +203 D /        ALL       VSAM D-disk
      DEVGUIDE +203 * /        ALL       I/S Development Guide
      DEVSQL   +222 Q /        ALL       SQL version 3.1 for IMF development
      DFSORT   =    = /        ALL       DFSORT Version 2.1
      DMS      +200 * /        ALL       DMS Version 2 Release 1
1= Help       2= Refresh  3= Quit      4= Sort(name)  5= Link        6= Alt PF
7= Backward   8= Forward  9= Category 10= Detach     11= Filelist   12= Cursor

====>
                                                    X E D I T  1 File
```

Figure 129. VMLINK Panel Interface

As shown in Figure 129, the VMLINK panel menu is self-explanatory. Also, some functions are available on the PF Keys, as shown in Table 16:

Table 16 (Page 1 of 2). VMLINK - Set 1 PF Keys Assigned by PROFVMLK XEDIT

Key	Setting	Action
Enter	EXECUTE	Execute commands typed on file lines or on the command line.
PF 1	Help	Display VMLINK command description
PF 2	Refresh	Update the list to indicate new or changed nickname entries, using the same parameters as those specified when VMLINK was invoked.
PF 3	Quit	Exit from VMLINK.
PF 4	Sort (name)	Sort by nickname.

Table 16 (Page 2 of 2). VMLINK - Set 1 PF Keys Assigned by PROFVMLK XEDIT		
Key	**Setting**	**Action**
PF 5	Link	Link to the disk using the vaddr and mode specified on the menu. If none specified, the defaults will be used.
PF 6	Alt PF	Toggle to PF key set 2.
PF 7	Backward	Scroll back one screen.
PF 8	Forward	Scroll forward one screen.
PF 9	Category	Display an XEDIT panel of category names.
PF 10	Detach	Detach the disk from your virtual machine.
PF 11	Filelist	Link the disk and issue the FILELIST command for all the files on the first disk. The disks will be restored when exiting FILELIST.
PF 12	Cursor	If the cursor is in the file area, move it to the command line. If the cursor is on the command line, move it back to the previous location in the file (or to the current line).

Notice that PF 6 key toggles to a second set of PF Keys. In Figure 130, the lower portion of the panel shows the second set of PF keys.

```
      DEVSQL   +222 Q /        ALL       SQL version 3.1 for IMF development
      DFSORT   =    = /        ALL       DFSORT Version 2.1
      DMS      +200 * /        ALL       DMS Version 2 Release 1
1= Help       2= Refresh  3= Quit      4= Sort(desc)  5= Sort(cat)   6= Alt PF
7= Backward   8= Forward  9= Autolink 10= Remove/A   11= Filelist   12= Cursor
```

Figure 130. Second VMLINK Panel Interface

The second set of PF keys are defined as follows:

Table 17 (Page 1 of 2). VMLINK - Set 2 PF Keys Assigned by PROFVMLK XEDIT		
Key	**Setting**	**Action**
Enter	EXECUTE	Execute commands typed on file lines or on the command line.
PF 1	Help	Display VMLINK command description
PF 2	Refresh	Update the list to indicate new or changed nickname entries, using the same parameters as those specified when VMLINK was invoked.
PF 3	Quit	Exit from VMLINK.
PF 4	Sort (desc)	Sort by description.
PF 5	Sort (cat)	Sort by category.
PF 6	Alt PF	Toggle to PF key set 1.
PF 7	Backward	Scroll back one screen.
PF 8	Forward	Scroll forward one screen.
PF 9	Autolink	Establish an autolink for the selected disk. The autolink is saved in the LASTING GLOBALV file.
PF 10	Remove/A	Remove an autolink for the selected disk.

Table 17 (Page 2 of 2). VMLINK - Set 2 PF Keys Assigned by PROFVMLK XEDIT		
Key	**Setting**	**Action**
PF 11	Filelist	Link the disk and issue the FILELIST command for all the files on the first disk. The disks will be restored when exiting FILELIST.
PF 12	Cursor	If the cursor is in the file area, move it to the command line. If the cursor is on the command line, move it back to the previous location in the file (or to the current line).

In addition to the above PF key definitions, the PROFVMLK XEDIT macro sets synonyms that can be used to sort the VMLINK nicknames. The synonyms are:

Synonym	Description
SNAME	Sorts the list alphabetically by nickname.
SCAT	Sorts the list alphabetically by category name.
SDESC	Sorts the list alphabetically by description.
SVDEV	Sorts the list in descending order by virtual device number.
SMODE	Sorts the list in descending order by filemode access.
SLMODE	Sorts the list in descending order by link address.

Returning now to Figure 129 on page 187, the PF 9 Key (as shown in Table 16 on page 187), displays a new panel interface, the *category* panel, as shown in Figure 131. This example is not very robust, since we have configured only two categories for the products, ALL and OFFICE. GRAPHICS, NEWS, TOOLS, and FORUMS are ideas for additional category names.

```
 VERONICA CATLIST  A0  V 80  Trunc=80 Size=1 Line=1 Col=1 Alt=0
Category Name
ALL
OFFICE

1= Help       2= Refresh  3= Quit   4=             5=             6=
7= Backward   8= Forward  9=       10=            11= VMLINK     12= Cursor
```

Figure 131. Category VMLINK Panel Interface

The category display has a new set of PF keys, defined for you in Table 18:

Table 18 (Page 1 of 2). VMLINK - Set 3 PF Keys Assigned by PROFVMLK XEDIT		
Key	**Setting**	**Action**
PF 1	Help	Display VMLINK command description.
PF 2	Refresh	Update the list to indicate new or changed nickname entries, using the same parameters as those specified when the category panel was entered.
PF 3	Quit	Exit from the category panel.
PF 4		Not assigned.
PF 5		Not assigned.
PF 6		Not assigned.
PF 7	Backward	Scroll back one screen.

Table 18 (Page 2 of 2). VMLINK - Set 3 PF Keys Assigned by PROFVMLK XEDIT		
Key	**Setting**	**Action**
PF 8	Forward	Scroll forward one screen.
PF 9		Not assigned.
PF 10		Not assigned.
PF 11	VMLINK	Issue a VMLINK for the selected category name.
PF 12	Cursor	If the cursor is in the file area, move it to the command line. If the cursor is on the command line, move it back to the previous location in the file (or to the current line).

F.2.2 Using VMLINK with Command Options

The command VMLINK, with options, offers a wide range of possibilities to gain access to all the needed information a virtual machine uses to perform its tasks.

When using the VMLINK command, it is necessary to remember how VMLINK default-overrides work:

1. The system VMLINK CONTROL setup is done by system support personnel in the VMLINK CONTROL control file.
2. A local VMLINK CONTROL file can be created by copying a VMLINK CONTROL from the system disk to a user minidisk. It overrides the system VMLINK CONTROL control file.
3. The nickname entries in the VMLINK NAMES file override the VMLINK defaults defined in a local or system VMLINK CONTROL control files.
4. The CMS DEFAULTS settings for default options to the VMLINK command.
5. Operands and options given in a VMLINK command override any defaults set up in a VMLINK NAMES file, local or system VMLINK CONTROL control file.

To show how all the overrides apply, see the examples shown in section “VMLINK Command Examples” on page 192.

NAMES File Exits Examples

To demonstrate how VMLINK handles the default overrides, simple PREEXITs, EXITs, and INVOKE EXECs are coded, run, and their console ouput is shown.

Discussed in section “VMLINK NAMES File Examples” on page 177, several exits were defined in the VMLINK NAMES file examples. The code for VMLINK PREXITB exit example is shown in Figure 132.

The code for VMLINK EXITB exit example is shown in Figure 133.

The code for VMLINK INVOKEB exit example is shown in Figure 134.

PREXITB EXEC Example

```
/*                                                                  */
/*     This is the PREXITB EXEC example that is specified on the    */
/*      preexit tag in the VMLINK NAMES file for PRODUCTB           */
/*                                                                  */

Address command
parse arg nick_id cat
say
say "I'm the PREXITB defined on the preexit tag of the NAMES file."
say "VMLINK passed me its category variable, .CA1.  So I got: "
say "Nickname ID : " nick_id
say "Category    : " cat
say
```

Figure 132. PREXITB EXEC Example

EXITB EXEC Example

```
/*                                                                  */
/*     This is the EXITB EXEC example that is specified on the      */
/*        exit tag in the VMLINK NAMES file for PRODUCTB            */
/*                                                                  */

Address command
parse upper arg nick_id vaddr filemode
say
say "I'm the EXITB defined on the exit tag of the NAMES file."
say "VMLINK passed me its virtual address variable, .CU1, and its"
say "filemode variable, .FM1.  So I got: "
say "Nickname ID     : " nick_id
say "Virtual Address : " vaddr
say "Filemode        : " filemode
say
```

Figure 133. EXITB EXEC Example

INVOKEB EXEC Example

```
/*                                                                  */
/*     THIS IS THE INVOKEB EXEC EXAMPLE THAT IS SPECIFIED ON THE    */
/*                VMLINK  NAMES File for PRODUCTB                   */
/*                                                                  */

Address command
parse arg nodes
say
say "I'm the INVOKEB defined on the invoke tag of the NAMES file."
say "I'm usually used to start up an application.   However, in this"
say "case, VMLINK passed me its node variable, .NO1.  So I got: "
say "Nodes : " nodes
say
```

Figure 134. INVOKEB EXEC Example

VMLINK CONTROL Control File Exits Examples

To see the code of the VMLINK PEXITCTL exit example defined in the VMLINK CONTROL control file, refer to Figure 135.

For the code of the VMLINK EXITCTL exit example defined in the VMLINK CONTROL control file, refer to Figure 136.

PEXITCTL EXEC Example

```
/*                                                                  */
/*     This is the PEXITCTL EXEC example that is used in the        */
/*                 VMLINK  CONTROL  File                            */
/*                                                                  */

say
say "I'm the PEXITCTL defined on the VMLINK CONTROL control file."
say "But the preexit tag on the NAMES file will override me."
say
```

Figure 135. PEXITCTL EXEC Example

EXITCTL EXEC Example

```
/*                                                                  */
/*     This is the EXITCTL EXEC example that is used in the        */
/*                 VMLINK  CONTROL  File                            */
/*                                                                  */

say
say "I'm the EXITCTL defined on the VMLINK CONTROL control file."
say "But the exit tag on the NAMES file will override me."
say
```

Figure 136. EXITCTL EXEC Example

VMLINK Command Examples

As mentioned in section F.2, “Using VMLINK” on page 185, before using the VMLINK command, you may require the VMLINK CONTROL and VMLINK NAMES files to be coded and activated by system support personnel. Advanced users can create their own. Of course, if you are using VMLINK with userid and disk or SFS options, you will not need NAMES or CONTROL files.

The examples in this section rely on the proper setup of the VMLINK CONTROL and VMLINK NAMES files. To show you VMLINK setup and usage, the examples in this section will use the VMLINK CONTROL file shown in Figure 116 on page 171, and the VMLINK NAMES file, MYOWN NAMES, shown in section “VMLINK NAMES File Examples” on page 177. While reading the following examples, please refer to the mentioned sections whenever needed.

Example 1:

This example shows the execution of the following command:

VMLINK PRODB

with the characteristics describe below:

- The entries declared for PRODUCT B as shown in Figure 122 on page 180 to Figure 125 on page 182, are in effect.
- The VMLINK CONTROL control file did not override any default values.

```
vmlink prodb

I'm the PREXITB defined on the preexit tag of the NAMES file.
VMLINK passed me its category variable, .CA1.  So i got:
Nickname ID :  PRODB
Category    :  PRODUCTS

DMSVML2060I Linking Product B Mdisk 191 ... linked RR as 500 file mode E

I'm the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
filemode variable, .FM1.  So I got:
Nickname ID     :  PRODB191
Virtual Address :  500
Filemode        :  E

DMSVML2060I Linking Product B Mdisk 193 ... linked RR as 902 file mode F

I'm the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
filemode variable, .FM1.  So I got:
Nickname ID     :  PRODB193
Virtual Address :  501
Filemode        :  F

DMSVML2060I Linking Product B Mdisk 195 ... linked RR as 903 file mode G

I'm the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
filemode variable, .FM1.  So I got:
Nickname ID     :  PRODB195
Virtual Address :  502
Filemode        :  G


I'm the INVOKEB defined on the invoke tag of the NAMES file.
I'm usually used to start up an application.   However, in this
case, VMLINK passed me its node variable, .NO1.  So I got:
Nodes :  TOTVM1 RMESNPM3

DMSVML2061I Linking Product B Mdisk 191 ... detached
DMSVML2061I Linking Product B Mdisk 193 ... detached
DMSVML2061I Linking Product B Mdisk 195 ... detached
Ready; T=0.15/0.19 15:05:27
sp console stop close
```

Figure 137. VMLINK PRODB with NAMES but without VMLINK CONTROL Activated

In Figure 137 the following steps can be seen:

1. The preexit EXEC, **PREXITB**, defined on the preexit tag of the NAMES file, was executed. It used the VMLINK variables passed to the EXEC as parameters: the category (.CA1).
2. VMLINK linked and accessed the PRODUCT B 191 minidisk with virtual address 500 and filemode E.
3. The exit EXEC, **EXITB**, defined on the exit tag of the NAMES file was executed for the preceding minidisk (191), showing the VMLINK variables passed to the EXEC as parameters: the virtual address (.CU1) and the filemode (.FM1).
4. VMLINK linked and accessed the PRODUCT B 193 minidisk with virtual address 501 and filemode F.
5. The exit EXEC, **EXITB**, defined on the exit tag of the NAMES file was executed for the preceding minidisk (193).
6. VMLINK linked and accessed the PRODUCT B 195 minidisk with virtual address 502 and filemode G.
7. The exit EXEC, **EXITB**, defined on the exit tag of the NAMES file was executed for the preceding minidisk (195).
8. The invoke EXEC, **INVOKEB**, defined on the invoke tag of the NAMES file was executed as the last step, showing the VMLINK variables passed to the EXEC as parameters: the valid nodes (.NO1).
9. All minidisks were detached because of the INVOKE tag. To keep the minidisks linked and accessed, use the KEEP option of the VMLINK command.

   Refer to section F.1, “VMLINK Implementation” on page 159 for more details on the VMLINK command.

Example 2:

The following example shows the execution of the following command:

VMLINK PRODB

with the characteristics describe below:

- The entries declared for PRODUCT B as shown in Figure 122 on page 180 to Figure 125 on page 182, are in effect.
- The VMLINK CONTROL control file shown in Figure 116 on page 171 is active.

```
vmlink prodb

I'm the PEXITCTL defined on the VMLINK CONTROL control file.
But the preexit tag on the NAMES file will override me.

I'm the PREXITB defined on the preexit tag of the NAMES file.
VMLINK passed me its category variable, .CA1.  So i got:
Nickname ID :  PRODB
Category    :  PRODUCTS

I'm the EXITCTL defined on the VMLINK CONTROL control file.
But the exit tag on the NAMES file will override me.

I'm the PEXITCTL defined on the VMLINK CONTROL control file.
But the preexit tag on the NAMES file will override me.

DMSVML2060I Linking Product B Mdisk 191 ... linked RR as 500 file mode E

I'm the EXITCTL defined on the VMLINK CONTROL control file.
But the exit tag on the NAMES file will override me.

I'm the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
filemode variable, .FM1.  So I got:
Nickname ID     :  PRODB191
Virtual Address :  500
Filemode        :  E

I'm the PEXITCTL defined on the VMLINK CONTROL control file.
But the preexit tag on the NAMES file will override me.

DMSVML2060I Linking Product B Mdisk 193 ... linked RR as 900 file mode F

I'm the EXITCTL defined on the VMLINK CONTROL control file.
But the exit tag on the NAMES file will override me.

I'm the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
filemode variable, .FM1.  So I got:
Nickname ID     :  PRODB193
Virtual Address :  501
Filemode        :  F

I'm the PEXITCTL defined on the VMLINK CONTROL control file.
But the preexit tag on the NAMES file will override me.

DMSVML2060I Linking Product B Mdisk 195 ... linked RR as 901 file mode G

I'm the EXITCTL defined on the VMLINK CONTROL control file.
But the exit tag on the NAMES file will override me.

I'm the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
filemode variable, .FM1.  So I got:
Nickname ID     :  PRODB195
Virtual Address :  502
Filemode        :  G

I'm the INVOKEB defined on the invoke tag of the NAMES file.
I'm usually used to start up an application.   However, in this
case, VMLINK passed me its node variable, .NO1.  So I got:
Nodes :  TOTVM1 RMESNPM6

DMSVML2061I Linking Product B Mdisk 191 ... detached
DMSVML2061I Linking Product B Mdisk 193 ... detached
DMSVML2061I Linking Product B Mdisk 195 ... detached
Ready; T=0.17/0.23 15:30:36
```

Figure 138. VMLINK PRODB with NAMES and VMLINK CONTROL Activated

In Figure 138 the following steps can be seen:

1. The following steps are followed by the resolution of **PRODB** nickname entry in the NAMES file:
   a. The preexit EXEC, **PEXITCTL**, defined on the VMLINK CONTROL control file was executed.
   b. The preexit EXEC, **PREXITB**, defined on the preexit tag of the NAMES file was executed, overriding the PEXITCTL exit defined on the VMLINK CONTROL control file.
   c. The exit EXEC, **EXITCTL**, defined on the VMLINK CONTROL control file was executed.
2. The following steps are followed by the resolution of **PRODB191** nickname entry in the NAMES file:
   a. The preexit EXEC, **PEXITCTL**, defined on the VMLINK CONTROL control file was executed.
   b. VMLINK linked and accessed the PRODUCT B 191 minidisk with virtual address 500 and filemode E.
   c. The exit EXEC, **EXITCTL**, defined on the VMLINK CONTROL control file was executed.
   d. The exit EXEC, **EXITB**, defined on the exit tag of the NAMES file was executed, overriding the EXITCTL exit defined in the VMLINK CONTROL control file.
3. The following steps are followed by the resolution of **PRODB193** nickname entry in the NAMES file:
   a. The preexit EXEC, **PEXITCTL**, defined in the VMLINK CONTROL control file was executed.
   b. VMLINK linked and accessed the PRODUCT B 193 minidisk with virtual address 501 and filemode F.
   c. The exit EXEC, **EXITCTL**, defined in the VMLINK CONTROL control file was executed.
   d. The exit EXEC, **EXITB**, defined on the exit tag of the NAMES file was executed, overriding the EXITCTL exit defined in the VMLINK CONTROL control file.
4. The following steps are followed by the resolution of **PRODB195** nickname entry in the NAMES file:
   a. The preexit EXEC, **PEXITCTL**, defined in the VMLINK CONTROL control file was executed.
   b. VMLINK linked and accessed the PRODUCT B 195 minidisk with virtual address 502 and filemode G.
   c. The exit EXEC, **EXITCTL**, defined in the VMLINK CONTROL control file was executed.
   d. The exit EXEC, **EXITB**, defined in the exit tag of the NAMES file was executed, overriding the EXITCTL exit defined in the VMLINK CONTROL control file.
5. The invoke exit, **INVOKEB**, defined on the invoke tag of the NAMES file was executed for **PRODB** nickname entry resolution as the last step in the VMLINK cycle.

6. All minidisks were detached because of the INVOKE tag. To keep the minidisks linked and accessed, use the KEEP option of the VMLINK command.

   Refer to section F.1, “VMLINK Implementation” on page 159 for more details on the VMLINK command.

Example 3:

The following example shows the execution of the following command:

VMLINK PRODA (KEEP INVOKE EXEC SAYHI

with the characteristics described below:

- The entries declared for PRODUCT A, as shown in Figure 120 on page 178 and Figure 121 on page 179, are in effect, as are the entries for PRODUCT C as shown in Figure 126 on page 183 to Figure 128 on page 184.
- This example is to show how the co-requisite invocation EXEC defined in a NAMES entry is treated. **No** VMLINK CONTROL control file was activated in order to keep the output of the VMLINK command as clear as possible.
- The **SAYHI EXEC** is a simple EXEC (refer to Figure 140) intended just to demonstrate the activation of the VMLINK INVOKE exit through the VMLINK command.

```
vmlink proda ( keep invoke exec sayhi

DMSVML2060I Linking Product A Mdisk 191 ... linked MR as 800 file mode G
DMSVML2060I Linking Product C Mdisk 193 ... linked RR as 200 file mode X

Hi there... Everything ok?

Ready; T=0.09/0.11 12:59:25
sp console stop close
```

Figure 139. VMLINK PRODA with Invoke EXEC

In Figure 139, the following steps can be seen:

1. VMLINK linked and accessed the PRODUCT A 191 minidisk with virtual address 800 and filemode G, as declared in its NAMES entry on Figure 121 on page 179.
2. VMLINK linked and accessed the PRODUCT C 193 minidisk with virtual address 200 and filemode X, as declared in its NAMES entry on Figure 128 on page 184.
3. No minidisks were detached because the option KEEP was used.
4. The SAYHI EXEC (see Figure 140 on page 198) was executed as the last task of this VMLINK command.

```
/*                                                                    */
/*     A simple exec to be used with the INVOKE option on the         */
/*                        VMLINK command.                             */
/*                                                                    */

say
say "Hi there... Everything ok?"
say
```

Figure 140. SAYHI EXEC Example

F.3 VMLINK Programming Interface

In addition to making accessing and releasing disks transparent for users and programmers, VMLINK has a rich programming interface with REXX EXECs:

- Data about the nicknames or about the accessed disks can be returned to the stack or in variables.
- Commands and routines can be invoked at different points in the accessing process (INVOKE, EXIT, and PREEXIT options and the corresponding tags).
- A call to VMLINK can specify parameters to be passed to commands and exit routines.
- A call to VMLINK can suppress tags in the nickname entry (for example, NOINVOKE, NOEXIT, and NOLIST options).
- Console messages can be suppressed (NOTYPE option).
- Accessing and releasing can be done without changing the prior configuration of the virtual machine (PUSH and POP options).
- VMLINK can search names files besides those listed in the VMLINK CONTROL file (*ADDFILE control statement).

Note: Given that VMLINK is an EXEC, within REXX EXECs VMLINK should be invoked under the CMS or command environments:

- Address cms 'VMLINK ...'
- Address command 'EXEC VMLINK ...'

There are three ways that VMLINK can pass data to a program:

- On the program stack
- In REXX stem variables
- As input parameters to a routine called during VMLINK processing.

Data can be identified by specifying VMLINK variables, the .MSG operand, and NAMES file tags in the options field. Variables can also be used in parameter lists.

F.3.1 Putting VMLINK Data on the Program Stack

To receive the messages generated by VMLINK in a REXX program, you can use the stack. Specify the .MSG option, and the messages will be stacked by default in FIFO order, prefixed with *.MSG. In our testing, the EMSG setting made no difference to the resulting message passed through the stack. When running in native CMS, setting EMSG to TEXT would remove the CMS message from the text.

```
type vmlstack exec

/* */
Address command
'VMLINK PRODLIST (.msg'
do queued()
  pull a
  say a
end

Ready; T=0.01/0.01 11:46:33
vmlstack
*.MSG DMSVML2060I LINKING PRODUCT D SFS DISK ... ACCESSED AS MODE K
*.MSG DMSVML2060I LINKING PRODUCT C MDISK 193 ... LINKED RR AS 500 FILE MODE L
Ready; T=0.10/0.12 11:46:39
```

You will note that in this example, the first disk was defined as a SFS directory, and therefore only accessed. Also, case translation of the stacked messages was due to a REXX PULL, not a PARSE PULL.

F.3.2 Putting VMLINK Variables in REXX Stem Variables

By adding the stem option to the VMLINK command, it is possible to assign data passed from VMLINK to a REXX stemmed-variable.

```
type vmlstem exec

/* */
Address command
'VMLINK PRODLIST (:product .msg stem stemvar.'
do i = 1 to stemvar.0
  say stemvar.i
end

Ready; T=0.01/0.01 12:34:06
vmlstem
.MSG DMSVML2060I Linking Product A SFS disk ... accessed as file mode x
.MSG DMSVML2060I Linking Product C Mdisk 193 ... linked RR as 122 file mode l
PRODLIST :PRODUCT
PRODA191 :PRODUCT .DIR VMSYSU:MAINT.SAMPLES
PRODC193 :PRODUCT PRODUCTC 193 -900 Z-A RR
Ready; T=0.08/0.09 12:34:10
```

From this example, we learn that even though PRODLIST is a list of products, VMLINK will still report a null :PRODUCT tag for it. Then, it will resolve the two products which were part of the list. Asking for additional tag values on the VMLINK command will result in an increase in the length of the stemmed-variable.

F.3.3 Passing VMLINK Variables as Parameters

We have already shown you how to pass VMLINK variables to exits (see section "VMLINK Command Examples" on page 192 for this information). It is possible to use the .PA tag in your names file to accept parms passed from the command line as well. Parms may be passed to the :EXIT, :INVOKE, or :PREEXIT tags in the names file. In the following example, the data PARM_LIST would be substituted where a .PA exists in the VMLINK names file.

```
VMLINK PRODLIST ( PARM PARM_LIST
```

F.4 VMLINK Autolink

VMLINK has the ability to link disks automatically when a user re-IPLs CMS. The standard SYSPROF EXEC was enhanced to execute 'EXEC VMLINK (AUTOLINK RUN' if a VMLINK EXEC exists on the system. Autolink RUN tells VMLINK to check the LASTING GLOBALV for any of its definitions set with the VMLINK (AUTOLINK SET command. Upon finding a definition, VMLINK performs the operation during the execution of the SYSPROF EXEC.

In the following example, you will see a simple demonstration of setting an autolink disk, IPLing CMS, then deleting the autolink disk.

```
vmlink .dir vmsysu:maint.samples < = J-H > ( autolink set
VMSYSU:MAINT.SAMPLES accessed as file mode J
Autolink status updated for VMSYSU:MAINT.SAMPLES
Ready; T=0.08/0.08 13:43:31
q search
MAINT   191   A    R/W
-       DIR   J    R/W   VMSYSU:MAINT.SAMPLES
MNT190  190   S    R/O
Y-DISK  19E   Y/S  R/O
Ready; T=0.01/0.01 13:43:36
i cms
VM/ESA REL 2.2, CMS 11 RSU9404

VMSYSU:MAINT.SAMPLES accessed as file mode J
Ready; T=0.12/0.14 13:43:40
vmlink .dir vmsysu:maint.samples < = J-H > ( autolink del
Autolink removed for .DIR VMSYSU:MAINT.SAMPLES
Ready; T=0.04/0.04 13:43:58
```

From the example, its important to see that VMLINK executed the link when setting the autolink. Also, the "=" sign was used as a place holder, so access modes J-H could be specified with a SFS disk.

List of Abbreviations

ABEND	abnormal end
ACF	availability control file
AIX/ESA	Advanced Interactive Executive/Enterprise Systems Architecture (IBM)
ALET	access list entry table
APAR	authorized program analysis report
API	application program interface
APPC/VM	Advanced Program-to-Program Communication/Virtual Machine (IBM)
AR	access register
ASCII	American National Standard Code for Information Interchange
ASM	assembler
BSG	Branch in Subspace Group
CACHE	high speed buffer
CCW	channel command word (System 360/370 architecture)
CFW	cache fast write (ESA)
CHK	check
CHPID	channel path id
CICS	Customer Information Control System (IBM)
CMS	Conversational Monitor System (VM-based software, IBM)
COLIS	Corporate On-Line Information Service (CMS-based software)
CON	console
CP	Control Program
CPU	central processing unit
CR	control register
CRC	cyclic redundancy check
CRR	Coordinated Resource Recovery (VM/ESA)
CS	communication services
CSE	Cross System Extension (a new VM/XA 2.0 function)
CSL	Callable Services Library (VM/CMS HLL-callable services)
CUIR	control-unit-initiated reconfiguration
DASD	Direct Access Storage Device
DCE	Distributed Computing Environment (OSF)
DCSS	discontiguous saved segment (VM/XA)
DDNAME	data definition name
DDR	DASD dump restore
DEL	delete
DFSORT	Data Facility Sort (IBM program product)
DFW	DASD fast write
DIR	directory
DIRMAINT	directory maintenance program
DITTO	data interfile transfer,testing, and operations utility
DMS	development (formerly: display) management system
DVF	dump viewing facility
EOS	end of service
EREP	Environmental Error Record Editing and printing program
ESA	Enterprise Systems Architecture (IBM)
ESCON	Enterprise Systems Connection (architecture, IBM System/390)
ESM	external security manager (VM/CMS SFS authorization exits)
EST	eastern standard time
EXEC	execution program
FBA	fixed block architecture
FIFO	first in/first out
FIPS	Federation of Information Processing Societies
FM	file mode (CMS term)
FN	file name (CMS term)
FST	file status table
FT	file type (CMS term)
GA	general availability
GCS	Group Control System (VM/VTAM)

GDDM Graphical Data Display Manager (IBM program product)

GDQF Graphical Display and Query Facility (IBM program product)

GUI graphical user interface

HX halt execution

I/O input/output

I/S information systems

IBM International Business Machines Corporation

ICKDSF Device Support Facilities (program product)

ICU Interactive Chart Utility (GDDM)

IDRC Improved Data Recording Capability (feature on 3480, standard 3490)

IEEE Institute of Electrical and Electronics Engineers

IPL initial program load

ISFC Inter-System Facility for Communications (VM/ESA)

ISO International Organization for Standardization

ISPF Interactive System Productivity Facility (MVS and VM)

ISPF/PDF Interactive System Productivity Facility/Program Development Facility

ITSO International Technical Support Organization

IUCV Inter-User Communication Vehicle

LAN local area network

LASP load address space parameters

LIFO last in/first out

LOGON log on (to sign on to the system)

LRECL logical record length

MACLIB macro library

MDC mini-disk cache/caching

MP multi processor

MUMPS/VM Massachusetts General Hospital utility multi programming system/virtual machine (IBM)

MVS Multiple Virtual Storage (IBM System 370 and 390)

NSS named saved system (VM/XA)

NVS non volatile storage

OEM original equipment manufacturer

OSF Open Software Foundation Inc. (IBM with DEC, Apollo, HP, Groupe Bull, Nixdorf, Siemens)

PC Personal Computer (IBM)

PMF Print Management Facility

PSW program status word

PTF Program Temporary Fix

PVM Pass through Virtual Machine

PW password

PWSCS Programmable Workstation Communication Services (for VM, IBM)

QMF Query Management Facility (IBM program product)

QMS Quality Management System

R/O read-only

R/W read/write

RDEV real device control block

RDR reader

RECFM record format

REL release

RPC Remote Procedure Call

RSCS Remote Spooling Communications Subsystem (VM′s counterpart to MVS JES NJE)

RSU Recommended Service Upgrade tape

SCP System Control Processor

SDWA System Diagnostic Work Area (MVS control block)

SES Serviceability Enhancement Stage (VM service installation tool)

SFS Shared File System (hierarchical sharable VM/CMS file system)

SMTP Simple Mail Transfer Protocol (Ethernet, based on TCP/IP)

SNA Systems Network Architecture (IBM)

SQL Structured Query Language

SVC supervisor call instruction (IBM System/360)

TCP/IP Transmission Control Protocol/Internet Protocol (USA, DoD, ARPANET; TCP=layer 4, IP=layer 3,

TOD time of day

TSAF Transparent Services Access Facility

US United States (country identifier, ISO standard 3166)

VM Virtual Machine (IBM System 370 and 390)

VM/ESA Virtual Machine/Enterprise Systems Architecture (IBM)

VM/SP Virtual Machine facility/System Product (IBM)

VMCF Virtual Machine Communication Facility (CP facility)

VMDBK Virtual Machine control block (31 bit address mode)

VMLIB VM library (system supplied VM/CMS CSL routines)

VMXA Virtual Machine/Extended Architecture

VSAM Virtual Storage Access Method (IBM)

VSE Virtual Storage Extended (IBM System/370)

VTAM Virtual Telecommunications Access Method (IBM) (runs under MVS, VM, and DOS/VSE)

VVT version vector table

WDFM withdrawn from market

WSC Washington Systems Center (IBM)

XA Extended Architecture

XCF Cross-System Coupling Facility (MVS)

XEDIT eXtended EDITor (the IBM VM system editor that allows text input and revision)

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

ITSO Technical Bulletin Evaluation RED000

VM/ESA Release 2.2 Overview and Usage Experiences

Publication No. GG24-4219-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

ITSO Technical Bulletin Evaluation **RED000**
GG24-4219-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

GG24-4219-00