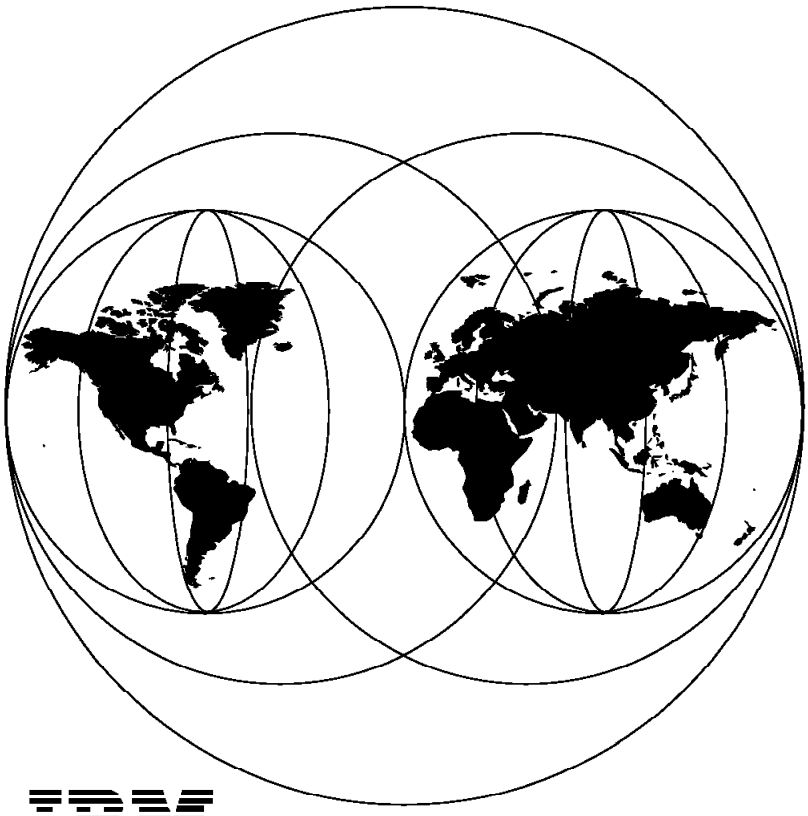
International Technical Support Organization

SG24-4584-00

MVS/ESA SP Version 5
Implementation Guide
Version 5.1.0
Version 5.2.0

November 1995



IBM

International Technical Support Organization
Poughkeepsie Center

International Technical Support Organization

SG24-4584-00

MVS/ESA SP Version 5
Implementation Guide
Version 5.1.0
Version 5.2.0

November 1995

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 (November 1995)

This edition applies to Version 5 Release 2 of MVS/ESA System Product (5655-068 or 5655-069).

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. 541 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Abstract

This document describes the new functions and enhancements of MVS/ESA Version 5 Release 2.0. It discusses all the items to consider when implementing the new functions introduced by MVS/ESA SP 5.2.0. The objective is to familiarize you with the new functions available and to give you some practical information on how to use these new capabilities.

This redbook discusses the many functions in MVS/ESA SP Version 5 that address installation complexities and make easier the task of systems management. These functions include: symbolic substitutions, dynamic subsystem interface, shared master catalog, job support for started tasks, I/O Path validation, hardware configuration definition enhancements, 4-digit device support, display RESERVE status, and dynamic exits.

In addition, the following topics are discussed: all the enhancements to problem management and dump management, a new MVS facility, subspaces, that provides a means of limiting the application server address space storage that an application program can reference, a new MVS function that permits more than one virtual storage page to simultaneously share the same system storage resource, support of relief for below 16MB virtual storage constraint by allowing the UCB to be defined in 31-bit storage above 16MB, and enhancements to APPC/MVS which have been implemented in MVS/ESA SP Version 5. Also described is the installation process and basic customization of OpenEdition MVS.

This redbook was written for customer system programmers and IBM technical personnel working in support of MVS/ESA environments. Some knowledge of MVS/ESA is assumed.

(224 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help IBM technical professionals and customers implement the new functions of MVS/ESA SP 5.1.0 and MVS/ESA SP 5.2.0. The information in this publication is not intended as the specification of any programming interfaces that are provided by MVS/ESA SP 5.1.0 and MVS/ESA SP 5.2.0. See the PUBLICATIONS section of the IBM Programming Announcement for MVS/ESA SP 5.2.0 for more information about what publications are considered to be product documentation.

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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. 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.

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 are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ACF/VTAM	CICS/ESA
DB2	DFSMS

DFSMS/MVS
ESA/390
IBM
MVS/ESA
RMF
ES/9000
ESCON
IMS/ESA
OpenEdition
VTAM

The following terms in this publication are trademarks of other companies:

Network File System, NFS — Sun Microsystems, Inc.

Windows is a trademark of Microsoft Corporation.

PC Direct is a trademark of Ziff Communication company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

Preface

This document is a guide for installing and using MVS/ESA SP Version 5. This document discusses all the items to consider when implementing the new functions introduced by MVS/ESA SP 5.1.0 and MVS/ESA SP 5.2.0. The objective is to familiarize you with the new functions available and to give you some practical information on how to use these new capabilities.

This document is intended for customers and IBM technical personnel responsible for installing and implementing the MVS operating system and its related components.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Introduction” gives a brief look at the new functions and enhancements made in MVS/ESA SP 5.1.0 and MVS/ESA SP 5.2.0
- Chapter 2, “Installing MVS/ESA SP Version 5” provides information to assist you in installing MVS/ESA SP 5.2.0. It identifies the new functions which require specific levels of hardware or software. Also discussed are the installation and migration requirements and changes to SYS1.PARMLIB.
- Chapter 3, “Exploitation of a Multisystem Environment” discusses those functions in MVS/ESA SP 5.1.0 and MVS/ESA SP 5.2.0 that address the complexities of managing a multisystem environment.
- Chapter 4, “System Management Improvements” describes improvements in MVS/ESA SP 5.1.0 and MVS/ESA SP 5.2.0 related to systems management.
- Chapter 5, “MVS Subspaces” describes a new MVS facility, subspaces. Subspaces provide a means of limiting the application server address space storage that an application program can reference.
- Chapter 6, “Dispatcher Enclaves” describes a new MVS/ESA SP 5.2.0 function that allows service requests to be controlled and reported with independence of the address space.
- Chapter 7, “Shared Pages” describes the MVS function that permits more than one virtual storage page can simultaneously share the same system storage resource.
- Chapter 8, “UCB Virtual Storage Constraint Relief” this support provides relief for below 16MB virtual storage constraint by allowing the UCB to be defined in 31-bit storage above 16MB.
- Chapter 9, “APPC/MVS Enhancements” describes the enhancements to APPC/MVS which have been implemented in MVS/ESA SP Version 5.
- Chapter 10, “OpenEdition MVS” describes the installation process and basic customization of OpenEdition MVS.
- Appendix A, “A Sample Program Utilizing the IOSPTHV Macro” has a program that prompts the user for a device number and returns path related information associated with the device number.
- Appendix B, “Sample Program for Subspace Management” has an example program to create, use, and delete a subspace.

- Appendix C, “Coding Sample for Enclave Implementation” has a simple program that uses the main macros for enclave handling.
- Appendix E, “Code Examples for APPC/MVS Enhancements” shows a subroutine used to issue CPIC requests and a REXX EXEC to call ATBEES3.
- Appendix F, “OpenEdition MVS” has information relative to the OpenEdition MVS chapter 8.

Related Publications

The following document is a guide to installing and using a base sysplex as implemented in the MVS/ESA Version 4 operating system and its related components. This document provides the reader with information about important aspects of using the new features of the cross-system coupling facility.

Short Title	Title	Order Number
Sysplex Migration Guide	*MVS/ESA Sysplex Migration Guide*	GG24-3925

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

A publication whose order number begins with the prefix LY is available to IBM-licensed customers only.

- Introduction/Evaluation

Short Title	Title	Order Number
General Information	*MVS/ESA General Information*	GC28-1422
JES2 Introduction	*MVS/ESA JES2 Introduction*	GC28-1420
JES3 Introduction	*MVS/ESA JES3 Introduction*	GC28-1421

- Information Management

Short Title	Title	Order Number
Library Guide	*MVS/ESA Library Guide with JES2*	GC28-1423
Master Index	*MVS/ESA Master Index*	GC28-1425

- Conversion

Short Title	Title	Order Number
MVS/ESA SP V5 Conversion Notebook	*MVS/ESA Conversion Notebook*	GC28-1436
MVS/ESA SP V5 JES2 Migration Notebook	*MVS/ESA JES2 Migration Notebook*	GC28-1437
MVS/ESA SP V5 JES3 Conversion Notebook	*MVS/ESA JES3 Conversion Notebook*	GC28-1438

- Operations

Short Title	Title	Order Number
MVS/ESA SP V5 Planning: Operations	*MVS/ESA Planning: Operations*	GC28-1441
MVS/ESA SP V5 System Commands	*MVS/ESA System Commands*	GC28-1442

- I/O Configuration Management

Short Title	Title	Order Number
MVS/ESA SP V5 HCD: Planning	*MVS/ESA Hardware Configuration Definition: Planning*	GC28-1445
HCD: User's Guide	*MVS/ESA Hardware Configuration Definition: User's Guide*	SC33-6468

- Multisystem Configuration Management

Short Title	Title	Order Number
Sysplex Overview	*System/390 MVS Sysplex Overview: An Introduction to Data Sharing and Parallelism*	GC28-1208
Sysplex Systems Management	*System/390 MVS Sysplex Systems Management*	GC28-1209
Sysplex Hardware and Software Migration	*System/390 MVS Sysplex Hardware and Software Migration*	GC28-1210
Sysplex Application Migration	*System/390 MVS Sysplex Application Migration*	GC28-1211
MVS/ESA SP V5 Setting Up a Sysplex	*MVS/ESA Setting Up a Sysplex*	GC28-1449
MVS/ESA SP V5 Sysplex Services Guide	*MVS/ESA Programming: Sysplex Services Guide*	GC28-1495
MVS/ESA SP V5 Sysplex Services Reference	*MVS/ESA Programming: Sysplex Services Reference*	GC28-1496
MVS/ESA SP V5 Planning: Global Resource Serialization	*MVS/ESA Planning: Global Resource Serialization*	GC28-1450
ESCON Manager Planning Guide	*Planning for the Enterprise Systems Connection Manager Version 1 Release*	GC23-0423
ESCON Director Planning	*Planning for the 9032 Enterprise Systems Connection Director*	GA23-0364
ESCON Migration Planning	*ESCON: Planning for Migration*	GG66-3181
Sysplex Timer Planning	*Planning for the 9037 Sysplex Timer*	GA23-0365

- Initialization and Tuning

Short Title	Title	Order Number
MVS Initialization and Tuning Guide	*MVS/ESA Initialization and Tuning Guide*	SC28-1451
MVS Initialization and Tuning Reference	*MVS/ESA Initialization and Tuning Reference*	SC28-1452
WLM Planning	*MVS/ESA Planning: Workload Management*	GC28-1493

- Problem Determination and Recovery

Short Title	Title	Order Number
Recovery and Reconfiguration Guide	*MVS/ESA Recovery and Reconfiguration Guide*	GC28-1458

- Customization

Short Title	Title	Order Number
MVS/ESA SP V5 Installation Exits	*MVS/ESA Installation Exits*	SC28-1459
TSO Programming	*MVS TSO Programming*	GC28-1460

- Application Development

Short Title	Title	Order Number
Authorized Assembler Services Guide	*MVS/ESA Programming: Authorized Assembler Services Guide*	GC28-1467
Authorized Assembler Services Library	*MVS/ESA SP V5 Authorized Assembler Services Reference, Volumes 1-4*	GC28-1475, GC28-1476, GC28-1477, GC28-1478
Callable Services for High Level Languages	*MVS/ESA Programming: Callable Services for High-Level Languages*	GC28-1464
WLM Reference	*MVS/ESA Programming: Workload Management Services*	GC28-1494
Assembler Services Guide	*MVS/ESA Programming: Assembler Services Guide*	GC28-1466

- Messages and Codes

Short Title	Title	Order Number
MVS/ESA Message Library	*MVS/ESA System Messages Volumes 1 - 5*	GC28-1480, GC28-1481, GC28-1482, GC28-1483, GC28-1484
Dump Output Messages	*MVS/ESA Dump Output Messages*	GC28-1485
MVS System Codes	*MVS/ESA System Codes*	GC28-1486
Routing and Descriptor Codes	*MVS/ESA Routing and Descriptor Codes*	GC28-1487

- Diagnosis: Tools and Services

Short Title	Title	Order Number
MVS/ESA SP V5 Diagnosis: Procedures	*MVS/ESA Diagnosis: Procedures*	LY28-1844

Short Title	Title	Order Number
MVS/ESA SP V5 Diagnosis: Tools and Service Aids	*MVS/ESA Diagnosis: Tools and Service Aids*	LY28-1845
MVS/ESA SP V5 Diagnosis: Reference	*MVS/ESA Diagnosis: Reference*	LY28-1872
Data Areas	*MVS/ESA SP V5 Data Areas, Volumes 1-5*	LY28-1857, LY28-1858, LY28-1859, LY28-1860, LY28-1861
IPCS User's Guide	*MVS/ESA Interactive Problem Control System (IPCS) User's Guide*	GC28-1490
IPCS Commands	*MVS/ESA Interactive Problem Control System (IPCS) Commands*	GC28-1491

- RMF

Short Title	Title	Order Number
RMF GIM	*MVS/ESA Resource Measurement Facility Version 5 General Information*	GC33-6481
RMF User's Guide	*MVS/ESA Resource Measurement Facility Version 5 User's Guide*	GC33-6483
RMF Reports	*MVS/ESA Analyzing Resource Measurement Facility Version 5 Reports*	LY33-9178
RMF Messages and Codes	*MVS/ESA Resource Measurement Facility Version 5 Messages and Codes*	GC33-6484

- Systems Architecture and Hardware

Short Title	Title	Order Number
ESA/390 Principles of Operation	*Enterprise Systems Architecture/390 Principles of Operation*	SA22-7201
Common I/O Device Commands	*Enterprise Systems Architecture/390: Common I/O Device Commands*	SA22-7204
ESCON I/O Interface	*Enterprise systems Architecture/390: ESCON I/O Interface*	SA22-7202
Sysplex Timer Planning	*Planning for the 9037 Sysplex Timer*	GA23-0365

- Time Sharing Option Extensions (TSO/E)

Short Title	Title	Order Number
TSO/E V2 Command Reference	*TSO Extensions Version 2 Command Reference*	SC28-1881
TSO/E Customization	*TSO Extensions Version 2: Customization*	SC28-1872

- Other Products

Short Title	Title	Order Number
Sysplex Automation Guide	*Automation and Console Functions in a Sysplex Environment*	GG24-3854
High Availability and Automation Management	*High Availability and Automation Management*	GG24-3829
NetView Installation and Administration	*NetView Installation and Administration Guide (MVS)*	SC31-7084
NetView Administration Reference	*NetView Administration Reference*	SC31-7080
DFSMShsm Implementation and Customization Guide	*DFSMS/MVS DFSMShsm Implementation and Customization Guide*	SH21-1078
DFSMSdss Storage Administration Reference	*DFSMS/MVS DFSMSdss Storage Administration Reference*	SC26-4929
DFSMSdss Storage Administration Guide	*DFSMS/MVS DFSMSdss Storage Administration Guide*	SC26-4930
VTAM V4R2 Network Implementation Guide	*VTAM V4R2 Network Implementation Guide*	SC31-6494
VTAM V4R2 Programming	*VTAM V4R2 Programming*	SC31-6496
VTAM V4R2 Resource Definition Reference	*VTAM V4R2 Resource Definition Reference*	SC31-6498

- OpenEdition MVS Related Publications

Short Title	Title	Order Number
OpenEdition MVS Command Reference	*MVS/ESA OpenEdition MVS Command Reference*	SC23-3014
C/370 Library Reference	*IBM SAA AD/Cycle C/370 Library Reference*	SC09-1761
OpenEdition MVS Shell and Debugger Messages	*MVS/ESA OpenEdition MVS Shell and Debugger Messages*	SC23-3780
Assembler Callable Services for OpenEdition MVS	*MVS/ESA Application Development Reference: Assembler Callable Services for OpenEdition MVS*	SC23-3020
LE/370 Debugging Guide and Run-Time Messages	*IBM SAA AD/Cycle Language Environment/370 Debugging Guide and Run-Time Messages*	SC26-4829
OpenEdition MVS Sockets	*IBM SAA AD/Cycle C/370 Library Reference: OpenEdition MVS Sockets*	SC23-3024
OpenEdition MVS Users Guide	*MVS/ESA SP V5 OpenEdition MVS Users Guide*	SC23-3013
IBM TCP/IP for MVS: User's Guide	*IBM TCP/IP for MVS: User's Guide*	SC31-7136

International Technical Support Organization Publications

The following redbooks contain information about the Version 5 sysplex environment. Books not already available will be published in 1996.

Short Title	Title	Order Number
JES Version 5 Overview	*MVS/ESA JES2 and JES3 Technical Announcement Presentation*	GG24-3330
JES2 Implementation	*MVS/ESA SP-JES2 Version 5 Implementation Guide*	SG24-4583
JES3 Implementation	*MVS/ESA SP-JES3 Version 5 Implementation Guide*	SG24-4582
MVS 5.1 Presentation Guide	*MVS/ESA 5.1.0 Technical Presentation Guide*	GG24-4137
RACF V2.1 Inst. and Impl.	*RACF V2.1 for MVS Presentation Guide*	GG24-4281
DFSMS/MVS 1.2.0 Parallel Sysplex Support	*DFSMS/MVS 1.2.0 S/390 Parallel Sysplex Support*	GG24-4400
WLM Performance Studies	*Workload Manager Performance Studies*	SG24-4352
WLM CICS Migration	*Migration to MVS V5 Workload Management in a CICS/VSAM Environment*	SG24-4353
Parallel Sysplex Perf.	*S/390 Parallel Sysplex Performance*	GG24-4356
Sysplex Guide	*MVS/ESA Version 5 Sysplex Migration Guide*	SG24-4581
MVS OpenEdition	*MVS/ESA SP 5.2.2 OpenEdition Initialization and Customization*	SG24-4529
RACF V2.1 Inst. and Impl.	*RACF V2R1 Installation and Implementation Guide*	GG24-4405
HCD and Dynamic I/O Reconfiguration Primer	*MVS/ESA HCD and Dynamic I/O Reconfiguration Primer*	SG24-4598

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

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

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

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

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books 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 local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

Acknowledgments

The advisor for this project was:

Bob Haimowitz and Paul Rogers International Technical Support Organization, Poughkeepsie Center

The authors of this document are:

Dave Clitherow	IBM United Kingdom
Bob Cogan	IBM Australia
Marcelo Corral	IBM Argentina
Juergen Dirksen	IBM Germany
Francois Gizard	IBM France
Martin Harker	IBM Australia
Hans Dieter Mertiens	IBM Germany
Andre van Wyk	IBM South Africa
Ralph H Rudd	IBM South Africa

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

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Kershaw Mehta	IBM OpenEdition MVS Development, Kingston
Graeme Port	Open Software Associates Ltd, P.O.Box 401, Ringwood, Victoria 3134, Australia
Dave Follis	APPC Development Poughkeepsie USA

Chapter 1. Introduction

This chapter provides an overview of the functions and enhancements available in MVS/ESA SP Version 5; they are discussed in greater detail in the following chapters. The enhanced sysplex functions, introduced in MVS/ESA SP Version 5, are not discussed in this document; they are described in *MVS/ESA Version 5 Sysplex Migration Guide*.

The functions and enhancements in MVS/ESA SP Version 5 are geared towards exploiting a multisystem environment as well as making system management tasks easier to perform. This document focus on the the installation, migration, and customization of MVS/ESA SP Version 5; the following areas are covered:

- Installation of MVS/ESA SP Version 5
- Exploitation of a multisystem environment
- Improvements in systems management
- Workload management simplification
- Subspace group facility
- Dispatcher enclaves
- Shared pages
- UCB virtual storage constraint relief
- APPC/MVS enhancements
- OpenEdition MVS enhancements

In addition, the following functions and subsystems have been enhanced:

- ESCON Manager and ESCON Manager interface
- I/O supervisor management
- Dispatcher restructure
- Enhanced processor system services
- RMF
- JES2 and JES3
- RACF
- SDSF

1.1 Installation of MVS/ESA SP Version 5

The installation of MVS/ESA SP Version 5 has specific requirements beyond the ones for MVS/ESA SP Version 4. One of them and, perhaps, the most important one, is the fact that with MVS/ESA SP Version 5 the use of hardware configuration definition (HCD) is mandatory. This means that to define your MVS configuration you must use HCD; you cannot use the MVSCP process. There is, however, with HCD, the capability of importing your MVSCP deck and creating the corresponding configuration.

The are also some functions that require specific levels of hardware and software to be exploited. See 2.1, “Hardware Requirements” on page 19, for details.

1.2 Exploitation of a Multisystem Environment

A sysplex is a set of MVS/ESA systems that communicate and cooperate with each other, using the applicable hardware and software, to process work. A significant difference between the sysplex and the conventional MVS/ESA system is the improved growth potential and level of availability that it offers. For this multisystem environment, which allows the spreading of work to grow beyond a single system, the following enhancements in MVS/ESA SP Version 5 were introduced:

- Coupling facility
- Cross-system extended services (XES)
- Sysplex failure management
- Enhanced event notification facility
- Serviceability enhancements
- Operator commands and operations services
- Sysplex device drain
- Sysplex GRS enhancements
- Automation of IOS messages
- SMF enhancements

1.2.1 Coupling Facility

IBM′s coupling technology makes high performance data sharing possible through a combination of hardware facilities and MVS/ESA SP Version 5 software services. The hardware consists of a coupling facility and high bandwidth fiber optic links called coupling facility channels.

A coupling facility is a special logical partition (LPAR) defined on a ES/9000 711-based machine, an LPAR on a S/390 9672, or an LPAR on a S/390 9674. Enhancements to PR/SM LPAR allow this special logical partition to be defined that runs the coupling facility control code (CFCC). This control code is IBM Licensed Internal Code (LIC).

The coupling facility channels are S/390 channels that use high bandwidth fiber optic cables to provide fast connectivity between the coupling facility and the connected systems. These links do not use the standard I/O channel protocol. Instead, they operate at much faster speeds (50 or 100MB/s) and are bidirectional; that is, sending and receiving can take place simultaneously.

Storage in the coupling facility can be allocated as three different types of named objects called structures. Each structure has specific attributes, assigned by the application that uses the structure:

List structures Used to maintain global information in lists or queues.

Cache structures Used to store data within the coupling facility storage.

Lock structures Used to serialize access to resources.

The application manipulates the structure or data in the structure by issuing one of a set of authorized macros services available through cross-system extended services (XES).

1.2.2 Cross-System Extended Services (XES)

The concept of a basic sysplex was introduced in MVS/ESA SP Version 4. MVS/ESA SP 5.1.0 expands the basic sysplex concept introducing the concept of a parallel sysplex. Cross-system coupling facility (XCF) is the component of MVS/ESA SP Version 4 that supports basic sysplex; it is a set of standard system interface and services that are available to authorized applications to allow communication between these application in a multisystem environment.

XES is an extension to the cross-system coupling facility component. XES provides services to authorized applications, such as subsystems and MVS components, to use the coupling facility to cache data, exchange status, and access sysplex lock structures so that techniques conducive to high performance data sharing and fast failure recovery can be implemented. XES services provide the following capabilities:

- High performance data sharing across the sysplex
- Maintenance of the integrity and consistency of shared data
- Sysplex high availability

1.2.3 Sysplex Failure Management

Sysplex failure management (SFM), a set of functions provide by XES to enhance the availability of a parallel sysplex, allows an installation to predefine the actions that MVS is to take when certain types of failures occur in the sysplex. For example, if one system loses signalling connectivity to other systems in the sysplex, SFM provides the method of reconfiguring the sysplex so that operations can continue. SFM allows an installation to specify a relative value to each system in the sysplex. Therefore, in the event of a connectivity failure, systems in the sysplex that are most important to the installation can continue without operator intervention.

SFM also supports the Processor Resource/Systems Manager (PR/SM) automatic reconfiguration facility. It allows the installation to specify the actions that are to be taken before a system is partitioned out of the sysplex.

1.2.4 Event Notification Facility Enhancements

The event notification facility (ENF) provides a method for MVS to signal the occurrence of special events to users that have identified themselves as listeners for the event. The notification occurs by scheduling the execution of an exit routine as established by the user.

The enhancements to the event notification facility increase the availability of data to the listener user exit routine. Listeners are able to:

- Pass parameters to the listener user exit routine when it gets control
- Request that the listener user exit routine get control in SRB mode in the listeners address space

Further enhancements simplify the end-of-task and end-of-memory cleanup processing for the listener. Event codes have been provided for coupling facility availability, WLM events, and SRM events.

Information on the enhanced facilities can be found in *MVS/ESA SP V5 Auth Assembler Services Reference ENF-ITT*.

1.2.5 Serviceability Enhancements

Enhancements to MVS/ESA SP Version 5 have been made in the area of production and management of dumps and traces. The enhancements help the installation to manage a sysplex or parallel sysplex environment with less effort, and bring benefits in people productivity and disk space saving.

Dump enhancements in MVS/ESA SP Version 5 continue the evolution towards the goal of a single-system image that is easy to manage, no matter how large or complex. The following areas were enhanced:

- Dump management: MVS/ESA SP V5.1 enables an installation to substantially increase the number of systems in a sysplex. This release ensures that dumps and traces are successfully produced and managed in a parallel sysplex.

  SVC dump supports the creation of dump data sets using names specified by the installation. By controlling the names of the dump data sets, an installation can assign meaningful names to identify specific dumps.

  SVC dump enhancements improve the utilization of direct access storage devices. The system automatically allocates dump data sets, of the proper size, at the time the dump is written to DASD. Automatic allocation of SVC dump data sets helps to minimize wasted DASD space and eliminates the need to determine:

  - How many dump data sets to allocate
  - What size to make them
  - The system with which to associate preallocated dump data sets
  - Which dump data sets need to be off-loaded and cleared

  Dump management is also enhanced by modifications to the DUMPDS and DISPLAY DUMP operator commands. These help to control and display the options for automatic allocation of dump data sets.

- IPCS: With MVS/ESA SP 5.1.0, IPCS improvements extend the number of traces that can be merged beyond 16 and address potential bottlenecks that could affect IPCS performance.

- Multisystem SVC dump enhancements: In a sysplex, an installation can request dumps from more than one system at a time to collect all of the data for a specific problem.

  To set up these dumps, an installation can use generic characters in system names, address space names, job names, data space names, user IDs, XCF group names, and XCF member names.

  When requesting dumps on other systems, the system provides an incident token for all of the dumps, so that system programmers can correlate the dumps to the problem.

- Stand-alone dump (SADUMP) enhancements:

  - Simplified migration of SADUMP to a new release of MVS.
  - Multi-volume support for DASD. This improves availability by helping to prevent truncated dumps.
  - Support for extended count key data (ECKD) architecture. Use of this architecture helps to improve performance by allowing non-synchronous I/O.

- Optimized blocksize for 3390 - IBM Direct Access Storage Devices, which helps improve dump efficiency.

- SLIP trap management: Changes in the SLIP command enable an installation to use a single SLIP command across systems, instead of an individualized SLIP command for each system. The improvements are:
  - Generic characters: An installation can use generic characters to identify a series of names that form a pattern.
  - Job names to identify data spaces: An installation can specify a data space using the job associated with it rather than the address space associated with it. Associating a job with a data space enables an installation to set SLIP traps before initializing the systems.
  - Automatic disabling of other traps for a problem: The installation can specify identical SLIP traps on more than one system in a sysplex. When the trap occurs and is disabled on one system, the other systems can automatically disable the remaining identical traps.
  - PER traps can be chained and indirect addressing is supported for the RANGE keyword.
- Dump analyses and elimination (DAE): DAE is enhanced to provide a better user interface and to make dump suppression more effective.
- Component trace: The trace has been enhanced to provide better data integrity and a more user friendly interface.

1.2.6 Operator Commands and Operations Services

The concept of sysplex and a sysplex-wide operation was introduced in MVS/ESA SP Version 4. This operational concept was the first step towards a single-system image implementation for operations. In MVS/ESA SP Version 5, this concept has been enhanced; enhanced functions introduced for operators made the operation of all systems in a sysplex as a single system more feasable.

The operator command enhancements as well as the operations services (replacement for console services) available in MVS/ESA SP Version 5 are briefly described bellow.

1.2.6.1 Operator Commands

The operator commands are enhanced to:

- Reduce operational complexity in a sysplex
- Avoid duplication of sysplex-wide command
- Provide operational usability and serviceability
- Remove operational constraints

These enhancements to sysplex operations reduce the complexity of managing a sysplex. Operators can treat the collection of systems in a sysplex as a single entity because of changes to command processing. They can cause most single-system commands to be processed on all systems in a sysplex (or on a named group of systems in the sysplex), and receive the collected responses at the issuing console. For a description of the operator command changes in MVS/ESA SP Version 5, see *MVS/ESA Version 5 Sysplex Operations Guide*.

1.2.6.2 Operations Services

MVS/ESA SP Version 5 includes a component trace function for operations services calles SYSOPS. This function provides increased capability for diagnostic and data capture both in a single or multisystem environment.

The SYSOPS component trace support includes:

- A CTIOPS00 PARMLIB member
- CTRACE parameter in the CONSOLxx PARMLIB member
- CT option on the TRACE operator command
- Dynamic sizing of trace buffers

MVS/ESA SP Version 5 also includes a HARDCOPY option on the OPERPARM parameter of the MCSOPER macro. This allows users of the MCSOPER macro to receive the MVS hardcopy message set. This facility can be used by applications to augment or replace the system log. By allowing the collection of all the messages in the MVS HARDCOPY message set from one or more systems in a sysplex, the MCSOPER macro can be used to create customized sysplex-wide system log applications.

Two terms must be defined:

Hardcopy message set The messages to be recorded on the log.

Log The installation-managed hardcopy medium where the hardcopy message set is recorded.

Messages in the hardcopy message set are command responses, broadcast messages, messages generated by compilers, and so on. The hardcopy message set is the set of messages that was formerly recorded only in the system log (SYSLOG).

In addition, MVS/ESA SP Version 5 provides a programming interface to vary device processing, which ESCON Manager Version 1 Release 3 uses to provide a sysplex-wide vary device online/offline function. MVS/ESA SP Version 5 reduces the need for an operator to do system-by-system operations.

MVS/ESA SP Version 5 also minimizes the number of messages issued during IPL for temporarily busy devices. This change decreases operator intervention at IPL.

1.2.7 Sysplex Device Drain

In releases prior to MVS/ESA SP Version 5, when a VARY OFFLINE command is issued for a device, the device is first placed in pending offline state, and goes offline only when all the jobs that allocated the device have deallocated the device. When a device is in a pending offline state, however, it is still eligible for certain allocation requests; so there can be a considerable delay between the time a VARY OFFLINE command is issued, and the time that the device goes offline.

In MVS/ESA SP Version 5, the vary offline process is favored over the requests to allocate the device. However, the operator and an installation exit are still given the chance to override the system′s decision, and let the allocation request be satisfied by allocating the pending offline device. This redesign is known as sysplex device drain. Sysplex device drain is a change in the vary offline processing design.

1.2.8 Sysplex Global Resource Serialization Enhancements

The global resource serialization enhancements in MVS/ESA SP Version 5 are:

- Global resource serialization support for 32 systems in a sysplex
- Changes to the RESMIL self-tuning algorithm

Global resource serialization has been enhanced to support a sysplex with up to 32 systems. As it is now possible to have a sysplex of up to 32 MVS/ESA systems, it is important that you have the functionality within MVS to control and manage such an installation.

If an installation uses global resource serialization to serialize access to global resources among units of work on multiple systems, the contents of the GRSCNFxx PARMLIB member are used during system initialization to define the global resource serialization complex. A global resource serialization complex consists of two or more systems connected by links. These links can be either CTCs managed by global resource serialization, CTCs managed by XCF, or XES structures. The systems in the complex use global resource serialization to serialize any shared resources. For example, global resource serialization controls access to data sets on shared DASD volumes at the data set level rather than at the volume level. (For more information on global resource serialization, see *MVS/ESA Planning: Global Resource Serialization*).

Prior to MVS/ESA SP 5.1.0, global resource serialization adjusted the actual time that the ring system authority (RSA) message resided at a system within a fixed range. In a large global resource serialization ring, the time taken to circulate the RSA message could become excessive.

Specifically, the self-tuning algorithm RESMIL is changed to take into account the number of systems in a global resource serialization ring.

The system programmer can control the trade-off between global ENQ response time and global resource serialization CPU utilization. This is done by specifying a value for the RESMIL keyword in the GRSCNFxx member of SYS1.PARMLIB.

For small systems, an additional value of *RESMIL(OFF)* is provided. You now specify RESMIL as follows:

```
GRSDEF RESMIL (minrestime)   or
GRSDEF RESMIL (OFF)
```

The number specified by RESMIL, in milliseconds, is the minimum RSA-message residency time (that is, the least amount of time that the RSA message is to spend in this system). If you specify OFF, then the system assumes a value of zero and performs no global resource serialization tuning. If you specify RESMIL(0), however, you allow global resource serialization to tune the RSA message residence in a range with a minimum value of zero milliseconds. If you omit the RESMIL keyword, the default value is ten milliseconds, and self-tuning is enabled.

1.2.9 Automation of IOS Messages

The design of the I/O validation process, at IPL time, has been changed since MVS/ESA SP 5.1.0. The major design changes was to eliminate the IOS120A and IEA120A messages that were unnecessarily issued for devices temporarily unavailable due to contention. Also, by performing the I/O validation in parallel on a group of devices, the validation of other devices is not delayed by contention or errors on a single device.

1.2.10 SMF Enhancements

The enhancements to SMF are in the following areas:

- SMF data set naming enhancements

  Data set naming enhancement removes the requirement that the names of SMF data sets include SYS1.MAN as a prefix and expands the maximum length of the data set name to 44 characters. The use of symbolic parameters is supported for both the SMF ID (SID) and the system name (from the SYSNAME parameter in IEASYSxx). Symbolic parameters is also supported for SMF data set names; its use simplifies the naming process. See 4.1, “Symbolic Substitution Overview” on page 77 for an explanation on symbolic parameters.

- SMF system ID enhancements

  SMF system ID enhancement allows an installation to define multiple SMF system identifiers in a single SMF parmlib member; therefore, an installation can use a single SMF parmlib member for the entire sysplex.

- Useability of the SMF DSNAME parmlib parameter

  The syntax definition is expanded to remove the naming restriction on the 36-data set limitation. Since a certain amount of storage is required for SMF to manage each of its data sets, the only real data set limitation is storage. Also, symbolic parameter substitutions in the data set name, by using SID and SYSNAME parmlib option values, is allowed.

More information on these enhancements can be found in 3.11, “SMF Enhancements” on page 74.

1.3 Systems Management Improvements

In this section we shall take a brief look at some of the enhancements to MVS/ESA SP Version 5 that affect the management of your MVS system environment. Most of the enhancements are in support of multisystem management, but this does not necessarily mean that they are not applicable to a non-sysplex environment. The enhancements we will take a brief look at are:

- Shared parmlib member support
- Shared master catalog
- 4-Digit device support
- Dynamic Subsystem Interface
- Job support for started tasks
- Display reserve status
- Dynamic exits
- HCD enhancements

Each of these topics is discussed in more detail later in this book. Additionally, these topics are also discussed in the *MVS/ESA SP V5 Conversion Notebook*.

1.3.1 Shared Parmlib Member Support

MVS/ESA SP 5.2.0 provides a set of system symbolics for use with many installation parameters. When these symbolics are used, a single installation definition (for example, a parmlib member) may be sharable among multiple systems. The principal restriction in previous releases of MVS was that some parameters had to be specific to certain systems. The solution is to support symbolic substitution in SYS1.PARMLIB member parameters; this allows the user to be able to use a single parmlib member to handle multiple instances throughout the sysplex, resulting in simplified system management and decreased system management cost. For further information, see 4.1, “Symbolic Substitution Overview” on page 77.

1.3.2 Shared Master Catalog

With MVS/ESA SP Version 5 it is now possible to share the master catalog between multiple systems. The principal restriction in previous releases of MVS was that some system data sets had fixed names, such as SYS1.LOGREC. MVS/ESA SP 5.1.0 introduces the concept of system name. The system name, specified in the IEASYSxx SYSNAME parameter, can be used as a symbolic name to provide unique data set names for SYS1.LOGREC and certain other system data sets. For examples see 4.3, “Shared Master Catalog” on page 87.

1.3.3 4-Digit Device Support

MVS/ESA SP 5.1.0 enables installations to use four hexadecimal digits when defining device numbers. This theoretically increases the maximum number of devices that can be defined to a single MVS image from 4,096 to 65,536. However, to limit the amount of virtual storage used below 16MB to the amount used in previous release, users can define approximately 1500 additional devices.

Four-digit device specification makes it easier for installations to use the same I/O definition files (IODFs) for multiple systems and provides more flexibility for unique device numbers across a sysplex. See 4.7, “4-Digit Device Support” on page 105 for more information.

1.3.4 Dynamic Subsystem Interface

Dynamic Subsystem Interface (SSI) allows installations to modify subsystems defined to the system without requiring an IPL. Previously, subsystems could only be defined at IPL through the IEFSSNxx parmlib member. The Dynamic SSI support includes a set of authorized system services that subsystems can invoke to:

- Define a new subsystem
- Activate a subsystem that was already defined
- Deactivate a subsystem that was already defined
- Store and retrieve subsystem-specific information
- Define subsystem options, which include deciding:
  - If a subsystem can respond to dynamic SSI commands
  - Under which subsystem a subsystem should be started

- Query subsystem information

Dynamic SSI provides the following benefits:

- Supports continuous operations by minimizing the need for users to IPL to add a new subsystem or to upgrade an existing subsystem
- Reduces service costs associated with modifying SSI control blocks by removing the need for subsystems to modify SSI control blocks and allowing a set of system services to make the necessary changes

1.3.5 Job Support for Started Tasks

MVS/ESA SP Version 5 introduced the ability to assign a jobname to a started task. This allows you to track multiple instances of the same started task more simply. Additionally, job level characteristics can be assigned to the started task to control output, specify accounting information and allow the use of SYSIN data. .

This capability allows users to associate job-related characteristics to a started task. For example, for a specific started task, users can provide more granular security. RACF Version 2 exploits job name assignment to provide dynamic security assignment, thereby avoiding IPLs needed to change the security of a started task.

Further details are in 4.4, “Job Support for Started Tasks” on page 91
.

1.3.6 Display Reserve Status

When a start pending missing interrupt (MIH) condition occurs in a multisystem environment (including non-MVS systems), MVS automatically displays the central processor serial number of a failed system that is holding a device reserve. If the failed system is in a sysplex, the system name is included in the display. This enables you to distinguish between MIH conditions caused by long-held reserves and other more complex configuration problems.

Using the information provided in the display reserve status message (IOS431I) and the accompanying ISG020I message, operators and system programmers are able to free the resource and reduce the duration of system outages.

The display reserve status message (IOS431I) is issued automatically. See 4.8, “Display RESERVE Status” on page 107 for more details.

1.3.7 Dynamic Exits

MVS/ESA SP 5.1.0 introduces a dynamic exits capability that provides system control of multiple exit routines called for an exit, and allows updates to exit processing without an IPL.

The SMF and allocation installation exits exploit this capability. Users can associate multiple exit routines with the SMF and allocation exits and control their use at IPL or while the system is running.

Users can also use the dynamic exits capability to define their own exits and control the use of those exits within a program.

The facility is controlled by the EXIT statement in the PROGxx parmlib member. As such, DFSMS/MVS is required to enable the use of the SET PROG command

to dynamically update the member in use. For more details refer to 4.9, “Dynamic Exits” on page 108.

1.3.8 HCD Enhancements

HCD provides a single point for I/O configuration definition and maintenance for a single processor or an entire sysplex. The HCD enhancements in MVS/ESA SP 5.1.0 includes:

Microprocessor Cluster Support: HCD provides support for the IBM 9672 processors having their service processors interconnected by a token-ring LAN in a S/390 microprocessor complex.

Coupling Facility Support: HCD provides support for coupling facility channels.

4-Digit Device Support: HCD code has been prepared to handle 4-digit device numbers since HCD Version 1 Release 1. The support in HCD consists of:

- A small enabling hit for the BCP
- An additional flag for those UIMs that support 4-digit devices
- The JES3 module shipped with HCD is upgraded to support 4-digit device numbers

Switch Configuration Enhancements: An ESCON Director (ESCD) allows, through its attached PS/2 system, the storing of several switch configurations (this is similar to having various IOCDS files stored for an ES/9000 processor).

With the ESCON Manager product, the user can retrieve one of these stored switch configurations and activate the switch data in the corresponding ESCON Director. Likewise, the user can save his switch configuration under a specific ID in the associated PS/2 ESCON Director.

HCD 5.1 has been enhanced to exploit these ESCON Manager (ESCM) functions. It allows the user to migrate switch configurations from an existing ESCD configuration file into HCD, and to store switch configurations from HCD into an ESCD configuration file.

MVSCP Elimination: MVS/ESA SP 5.1.0 no longer allows a configuration that was built by using MVSCP. HCD must be used to create an MVS I/O configuration.

Migration Enhancements: The “partial migration” function allows the user to update existing processor and operating system configuration definitions in an IODF with IOCP, MVSCP, or HCPRIO control statements. In particular, one can:

- Add or replace control units, devices, channel paths, and partitions
- Replace list of consoles
- Update EDTs, generics, and esoterics

Graphical Configuration Reports: This function allows the user to create or view various types of graphical configuration reports of the logical configurations stored in the IODF. The generated reports can then be printed by means of DCF/GML or BookMaster. Furthermore, an AFP printer, such as a 3820 or 3800, is required. To display a report, GDDM is needed. The printed output can be used for documentation purposes and serve as a base for further configuration

planning. The report can be limited by means of filtering. The display function allows the user to get a quick overview of the logical hardware configuration.

Usability Enhancements: Many usability enhancements have been included in HCD 5.1, including the capability to copy configuration data between IODFs, and the capability to change CHPID types.

HCD in MVS/ESA SP 5.2.0, has been enhanced to support the definition of UCBs above 16MB.

For additional information on the enhancements in HCD, refer to the *HCD User's Guide*.

1.4 Workload Management Simplification

MVS/ESA SP Version 5 introduces a goal-oriented management philosophy for MVS systems that allows an installation to state its system performance expectations in common business terminology generally used to document service level agreements (SLAs). A high level interface for specifying performance goals replaces the specification of low level resource-oriented MVS tuning parameters. MVS manages contention for resources and adapts dynamically as required. In instances of resource contention or insufficient capacity, MVS protects the work most critical to the installation without external intervention, improving the installation's ability to meet its business objectives. The benefits of this approach to workload management include:

- Fewer, simpler system externals
- Externals that reflect user expectations
- Expectations-to-feedback correlation
- System managed toward expectations

Workload manager allows users to move to an environment where the system manages itself toward installation goals. This allows the staff to more effectively manage its current workload or to grow the workload and the number and size of systems that they must manage.

1.4.1 Defining Performance Goals

An installation defines performance goals in a service policy. Service policies are defined through an ISPF application, and they set goals for all types of MVS-managed work, online and batch. An installation can create multiple service policies to adjust performance goals for different periods of time.

The service policy applies to a sysplex, whether a single system or a multisystem sysplex. Each service policy has a name, and can be activated by the workload manager (WLM) ISPF application, or an operator command. Only one policy can be active at a time. When the policy is activated, all systems in the sysplex process to achieve the goals defined in the policy.

Workload management coordinates and shares performance information across the sysplex. Using this information, each MVS system handles its own system resource management and dynamically optimizes resources to work according to the goals defined in the service policy. During processing, the system monitors how well the goals are being met across the sysplex and adapts accordingly as the environment changes. If there is contention for resources, each system makes the appropriate trade-offs based on the importance of the

work and how well the goals are being met. This way all systems can cooperate to protect work that is critical to your installation.

1.4.2 Reporting

Resource Measurement Facility (RMF) combines system management facilities (SMF) data for the sysplex, and reports how well the sysplex is doing to achieve the goals defined in the service policy. In addition, execution delay information is available in SMF records that show where delays are occurring. An installation can use this information to help adjust the performance goals, focus on specific subsystems having a problem, or make work scheduling adjustments.

1.4.3 Workload Distribution

Although workload management strives to achieve performance goals for all work in a sysplex, it remains the responsibility of the individual subsystems to distribute work across the sysplex to satisfy those goals. To make the most of workload management, work needs to be properly distributed across the sysplex and within each system. An installation must use the routing and distribution mechanisms of VTAM, TSO/E, CICS, IMS, JES and OpenEdition MVS to distribute work properly. CICSPlex System Manager (SM) is also available, offering a goal-oriented algorithm for dynamic transaction routing. For more information about CICSPlex SM, see *CICSPlex SM Concepts and Planning*.

Workload management replaces the existing set of SRM controls: the IEAICSxx and the IEAIPSxx parmlib members. However, with MVS/ESA SP Version 5, you have the option of processing with your existing IEAIPSxx and IEAICSxx members until you are comfortable wit the workload management service definition. Processing with an existing IEAIPSxx and IEAICSxx member is called compatibility mode. Processing towards goals defined in the workload management service definition is called goal mode. This way, you can switch over to processing towards a service policy according to your own pace.

For complete information about the workload management service definition, see *MVS/ESA SP V5 Planning: Workload Management*.

1.5 Subspace Group Facility

The subspace group facility has been developed to enhance the capabilities of service provider subsystems. This type of subsystem uses a single server program to provide common functions to multiple application programs (transactions) all running within a single address space. Until the introduction of the subspace group facility, one application program, in rare cases, could gain access to code and data belonging to another application. Accidental access to another application's code or data can result in application failures, service outages, and incorrect modification of critical business data. Problems resulting from these kinds of situations are generally costly and time consuming, and may require extensive recovery procedures.

With the subspace group facility, an application attempting to access storage outside of its own subspace is abnormally terminated. Users can easily identify applications that have failed or that would cause data integrity problems. Once an application (transaction) program is identified, resolving the problem is usually quite easy. Without this capability, it can be very difficult to identify programs that are causing storage violations and to resolve the problems.

Note: The subspace group facility is available on all IBM ES/9000 9021 711-based, 9121 511-based, 9221 211-based processors, and S/390 9672 processors. More detail on MVS Subspaces and how to use them can be found in Chapter 5, “MVS Subspaces” on page 117.

1.6 Dispatcher Enclaves

Enclaves are introduced with MVS/ESA SP 5.2.0 and provide the following functional enhancements:

- Allow selected service request (SRBs) to be identified, managed and reported on as a group or *enclave*. This support allows an installation to control the dispatching policy for SRBs grouped together as part of an enclave.
- Provide support for *preemptable* SRB routines, allowing work such as CPU-intensive or low-priority routines to execute asynchronously and to be scheduled without task-related overhead.
- Provide support for *client* SRB routines, allowing SRBs scheduled on behalf of a client to run in another address space but have its dispatching priority and CPU service time associated with the client address space.

1.7 Shared Pages

This MVS/ESA SP 5.2.0 function permits two address (or data) spaces to share the same real storage. This is useful in reducing the amount of storage used when two or more programs can share the same data. The shared pages function provides for copying-on-write (unique write). This permits a significant savings when the data to be shared is also written by more than one program, since only the written areas are duplicated.

Shared pages also reduces the amount of I/O performed when sharing large amounts of data. It provides a method of doing check pointing using processor storage, and communication between programs.

Macro services can define storage areas to be shared by programs within or between address spaces or data spaces. This can reduce the amount of processor storage required and the I/O necessary to support large data applications where sharing is required. It provides a way for programs, which must run below 16MB in 24-bit addressing mode, to access data residing above 16MB.

Refer to Chapter 7, “Shared Pages” on page 133 for more details.

1.8 UCB Virtual Storage Constraint Relief

MVS/ESA SP 5.2.0 relieves virtual storage constraints for installations that have a large number of I/O devices defined to MVS in a manner that preserves the existing access method interface. With MVS/ESA SP 5.2.0, installations can specify that a UCB reside above 16MB to conserve common virtual storage while maintaining compatibility with existing standard services.

The benefits of defining UCBs above 16MB include:

- Relieving constraints on below 16MB common virtual storage caused by UCBs by defining the UCBs above 16MB.
- In some cases, relieving constraints on private virtual storage by moving enough UCBs above 16MB and expanding the private storage area into the next 1 MB segment.

1.9 APPC/MVS Enhancements

The enhancements to APPC/MVS are as follows:

Diagnostic enhancements: MVS/ESA SP 5.1 increases the ability to diagnose errors in APPC/MVS transaction programs (TPs). APPC/MVS TPs can call a MVS TP conversation service to obtain detailed information about errors that occur in calls to MVS TP conversation services and common programming interface (CPI) communications calls. Programmers can use the returned information to diagnose and fix problems that occur when developing APPC/MVS TPs.

When APPC/MVS finds errors during processing of inbound requests, APPC/MVS can send error log information to a partner system. Programmers for partner systems can use the error log information to diagnose and fix problems in partner TPs.

Conversation Level Pacing: APPC/MVS now supports conversation-level pacing, which allows APPC/MVS TPs to send and receive large amounts of data without affecting the sending and receiving of data by other APPC/MVS TPs. APPC/MVS now controls the number of send and receive buffers that each conversation is using.

Conversation Correlator Enhancement: Alternate schedulers now have access to a *conversation correlator* value that a requestor can send as part of an inbound request. A partner TP sends a conversation correlator to the scheduler to uniquely identify a conversation. After the scheduler processes the request, the scheduler sends the correlator back to the partner TP. The partner TP can then correlate the original request with the response from the scheduler.

Using Common Area Data Spaces: APPC/MVS TPs can use SCOPE=COMMON data spaces (also known as common area data spaces) to make data accessible to all programs in a system.

More information on all of the APPC enhancements can be found in Chapter 9, “APPC/MVS Enhancements” on page 155.

1.10 OpenEdition MVS Enhancements

OpenEdition, initially introduced as a feature on MVS/ESA SP 4.3, is integrated into the MVS/ESA SP 5.1.0 product. Enhancements, such as integrated sockets support, along with TCP/IP modifications coupled with DCE and POSIX support, provides users and vendors with an operating system that makes it easier to port UNIX applications to MVS.

By integrating OSF/DCE components into MVS, IBM is expanding the role of MVS as an open system. DCE makes current data and logic available to heterogeneous clients by using a single set of standardized interfaces. By

supporting a comprehensive set of standard interfaces, OpenEdition DCE Base Services provides a de facto standard method for heterogeneous application interoperation.

OpenEdition DCE: OpenEdition DCE base services in MVS/ESA SP 5.1.0 supports:

- **Remote procedure call (RPC):** RPC provides substantial reductions in the amount of time required to code a distributed application and in the amount of code that must be written and maintained by the applications programmer. It lets the programmer concentrate on writing business logic required by an application by removing many of the network sensitivities sometime maintained within applications.
- **Directory Services:** By providing a single, distributed directory service it is no longer necessary to maintain local copies of information about resources. This results in much greater flexibility in the placement of logic and data in the network.
- **Security Services:** The security support ensures that clients and servers are trusted in a heterogeneous network. It also protects all resources from attack through the use of access control. This combination provides a cross-vendor, cross-platform security capability that is not found elsewhere.
- **Time Services:** The time services support a way to periodically synchronize the clocks on the different nodes in a distributed network, and a way of keeping the synchronized time in each node reasonably close to the correct time.

NFS Support in MVS OpenEdition: NFS is a Network File System that allows a workstation to access files that may reside elsewhere in a network as if they were resident on the workstation. Currently an NFS client can access MVS datasets using the NFS server (NFSS feature of DFSMS) support that IBM currently provides.

The NFSS feature along with support coded in MVS/ESA SP 5.1.0 allows the server to directly interface with the OpenEdition Hierarchical File System (HFS). Client applications or workstations can mount the OpenEdition/MVS HFS as a local file system and treat files as an extension of the workstation′s file system.

Integrated Sockets Support: Integrated sockets support is a sockets API that includes the functionality of BSD 4.3 UNIX sockets (Berkley University), integrated into the OpenEdition/MVS POSIX 1003.1 environment. It includes the following:

- A forkable thread safe and enabled sockets API that serves as a foundation for the porting or implementation of typical client-server functions found in UNIX systems
- A single runtime library to support a common API used by OpenEdition/MVS, OpenEdition MVS DCE, and sockets applications
- Support for UNIX-domain sockets for local interprocess communications
- Support consistent with eventual conformance to the evolving POSIX 1003.1G Networking standard defined sockets interface

Multiple Shell Sessions: OpenEdition MVS has been enhanced to allow users to start more than one shell session within the shell. Only one session is displayed on the screen. Additional OpenEdition MVS subcommands have been provided to enable ease of moving between sessions.

dbx Debugger Enhancements: The dbx debugger has been enhanced to provide support for objects known as threads, mutexes and condition variables.

Internationalization: Internationalization allows a program to run in different countries/language environments without any need for changes to the actual program.

Integrated LE/370: LE/370 functions are now included in the BCP.

REXX Support: REXX programs may now access OpenEdition MVS services and functions from MVS, TSO or the OpenEdition MVS environment.

RMF Enhancements: Additional SMF records allow comprehensive reporting of OpenEdition MVS usage.

MVS/ESA SP Version 5 Release 2.2 integrates into MVS all functions required for the XPG4 Base specification plus 90% of the functions defined by the XPG4.2 Single UNIX Specification, and provides additional support for interoperability. Some of the open systems function on the MVS/ESA platform are integrated into supporting products such as TCP/IP, VTAM, IMS, and so on.

Not only is there more function in the shell and utilities with the new XPG4 functions, but there are additional ways to access it and additional terminal types supported.

These new standardized UNIX interfaces that have been added to MVS/ESA SP Version 5 Release 2.2 will support the customer to:

- Port UNIX applications to MVS/ESA SP Version 5 Release 2.2
- Expand the scope of POSIX applications to include new UNIX functions
- Develop and use DCE client/server applications on MVS
- Reduce costs by attaching ASCII terminals to MVS/ESA SP Version 5 Release 2.2
- Support UNIX applications that need to share data in a heterogeneous network
- Use existing MVS data bases in a heterogeneous network

Since the application programming interfaces have been expanded to include significant UNIX function, users will have a broader range of applications to choose from that run on MVS/ESA SP Version 5 Release 2.2. This includes both traditional programming interfaces and the new UNIX interfaces.

The XPG4 and XPG4.2 support introduced in OpenEdition MVS/ESA SP 5.2.2 consists of:

- **Application Programming Interfaces (API′s):** XPG defines a set of system calls that are currently defined in terms of C Language bindings although they are partially accessible from other HLLs in MVS through the assembler interface. C runtime functions (C/C++ Language Support Feature) are available as part of the MVS/ESA SP 5.2.2 operating system.

- **UNIX Shell Interface:** MVS OpenEdition provides a XPG4-compliant shell interface, that has the look and feel of a UNIX system. Standard UNIX utilities are also provided. This interface and the utilities provide for portability of UNIX skills.
- **SNA Network for Socket Applications:** MVS/ESA SP 5.2.2 has made enhancements to the sockets interface, and in a subsequent release of ACF/VTAM, OpenEdition MVS/ESA SP 5.2.2 DCE applications and user written socket applications can use SNA networks transparently. This allows both TCP/IP and SNA networks to operate simultaneously for the same application.

  Sockets are a communications channel that enable unrelated processes (applications) to exchange data, whether the processes are on a single system or multiple systems.

  The MVS OpenEdition socket application environment is enhanced to support socket application transport over SNA networks via APAR OY10895 for the VTAM Version 4 Release 2 AnyNet/MVS feature. With this support the integrated sockets provided in MVS/ESA SP 5.1 can now scan over SNA or TCP/IP but not simultaneously from the same application.

  In the future, the VTAM AnyNet/MVS feature with the new OE converged sockets provided in MVS/ESA SP 5.2.2 will also support the socket application transport over SNA networks. However, this support will permit OpenEdition MVS socket applications to simultaneously run either SNA or TCP/IP networks.

Additional information on OpenEdition MVS can be found in Chapter 10, “OpenEdition MVS” on page 163.

Chapter 2. Installing MVS/ESA SP Version 5

This chapter provides information to assist you in installing MVS/ESA SP Version 5. New functions that require specific levels of hardware or software in order to be exploited are also identified. The following topics are discussed in this chapter:

- Hardware requirements
- Software requirements
- Installation and migration requirements
- Changes to SYS1.PARMLIB

2.1 Hardware Requirements

MVS/ESA SP Version 5 runs on any IBM processor that supports IBM Enterprise Systems Architecture/370 (ESA/370), namely:

- 9672 Parallel Transaction Server and 9672 Parallel Enterprise Server (all models)
- IBM Enterprise System/9000 (ES/9000) Processors:

  9021 Models:

  - 711-based models: 711, 821, 822, 831, 832, 941, 942, 952, 962, 972, 982, and 9X2
  - 520-based models: 520, 640, 660, 740, 820, 860 and 900
  - 340-based models: 330, 340, 500, 580, 620, and 720

  9121 Models:

  - 511-based models: 311, 411, 511, 521, 522, 621, 622, 732, and 742
  - 320-based models: 180, 190, 210, 260, 320, 440, 480, 490, 570, and 610

  9221 Models:

  - 170-based Models: 120, 130, 150, 170, 200

    Note: Previously referred to as 150-based models.
  - 211-based models: 191, 201, 211, 221, 421

  or equivalent, to maximize utilization of the Enterprise Systems Architecture/390 (ESA/390).

- IBM Enterprise System/3090 (ES/3090) Processor Model E at SEC 223670 or above; Model S at SEC 223770 or above; Model J; or model JH, to support ESA/370.

  The ES/3090 J-Models (Models 180J, 200J, 280J, 300J, 380J, 400J, 500J or 600J) must have:

  - SEC 227574 or above to support ESA/390 options ESCON and Sysplex Timer on ESA/370
  - SEC 227540 or above for the Move Page facility
  - SEC 227570 or above for Processor Resource/Systems Manager (PR/SM) Dynamic Storage Reconfiguration
  - SEC 227576 and RPQ to support the ESA/390 option Integrated Cryptographic Feature (ICRF) on ESA/370

  The ES/3090 Model S must have:

  - SEC 228862 or above for the Move Page facility
  - SEC 227570 for PR/SM Dynamic Storage Reconfiguration

- IBM Enterprise System/3090-9000T (ES/3090-9000T) Processor Models 15T, 17T, 18T, 25T, 28T, to support ESA/370 and provide some optional ESA/390 facilities.

  The ES/3090-9000T must have:

  - System engineering change (SEC) 227574 or above to support the ESA/390 options ESCON and Sysplex Timer
  - SEC 227574 or above for the Move Page facility
  - SEC 227574 or above for PR/SM Dynamic Storage Reconfiguration
  - SEC 227576 and request for price quotation (RPQ) to support the ESA/390 option Integrated Cryptographic Feature (ICRF)

- IBM Enterprise System/4381 (ES/4381) Model Group 90E, 91E, or 92E processor, which supports the IBM Enterprise Systems Architecture/370

The subspace group facility requires selected ES/9000 processors. These are:

- ES/9000 9021: 711-based models with SEC 228250 (minimum)
- ES/9000 9121: 511-based models with SEC C35954 (minimum)
- ES/9000 9221: 211-based models
- S/390 9672 Processors

Refer to *MVS/ESA Planning: Installation and Migration with JES2* for specific hardware requirements.

2.1.1 Coupled Systems Hardware Requirements

For MVS/ESA SP Version 5 to run in a coupled system environment, refer to *MVS/ESA Version 5 Sysplex Implementation Guide* and *MVS/ESA SP V5 Planning: Install and Migration JES2* or *MVS/ESA SP V5 Planning: Install and Migration JES3*, as appropriate. Beside the CPC, have your hardware service representative check your full hardware configuration to ensure all required microcode updates are installed.

2.1.2 Processor Storage Requirements

MVS/ESA SP Version 5 requires an increased amount of processor storage when compared to earlier versions. IBM bases the numbers in this book on experience with MVS/ESA SP V5.

Based upon IBM internal measurements, the minimum storage to IPL MVS/ESA systems is as follows:

- MVS/ESA SP 4.1 with MVS/DFP 3.2: 12 MB
- MVS/ESA SP 4.2, 4.2.2, or 4.3 with MVS/DFP 3.3: 14 MB
- MVS/ESA SP 4.3 with DFSMS/MVS 1.1: 16 MB
- MVS/ESA SP 5.1 with DFSMS/MVS 1.2: 20 MB
- MVS/ESA SP 5.2 with DFSMS/MVS 1.2: 22 MB

2.1.2.1 Storage Required for a Minimal Test Configuration

IBM recommends at least 24 megabytes to IPL a small test configuration in which the system programmer can do some limited work. A more realistic test environment requires 32 megabytes of central storage, plus an additional 24 or 32 megabytes (which can be central storage or expanded storage).

The minimal storage requirements depend on a number of factors, including:

- The number of devices in the I/O configuration
- The amount of storage configured
- The number of address spaces created
- The types of applications supported

2.1.3 DASD Requirements for Stand-Alone Dump to DASD

Starting with MVS/ESA SP 4.3, you can activate SADMP to DASD. If you do, you must allocate a SYS1.SADUMP data set that is large enough to contain the entire dump. This includes all central storage, all expanded storage, and all paged out data spaces. IBM recommends that you dedicate an entire volume to SADMP to DASD.

Note: If you do not allocate enough space for the dump, SADMP truncates the dump and supplies a message indicating how much storage was actually dumped.

For more information, see *MVS/ESA Diagnosis: Tools and Service Aids*.

2.2 Software Requirements

To check this software requirements refer to the appropriate announcement letters and to *MVS/ESA Planning: Installation and Migration with JES2*. For newest PTFs check PSP-Bucket.

Note: MVS/ESA SP Version 5 Release 1 is the last MVS release to support Assembler H. MVS/ESA SP Version 1 Release 2 requires High Level Assembler (5696-234).

2.2.1 Compatibility

MVS/ESA SP Version 5 is upwardly compatible with MVS/ESA SP 4.3.

2.2.2 JES2 and JES3 Considerations

The following table shows the levels of Job Entry Subsystem (JES) that are supported under MVS/ESA SP 5.2.0 BCP.

Table 1. Supported JES Levels. The levels of JES shown below are all supported under the MVS/ESA SP 5.2.0 BCP.

JES2	JES3
JES2 5.2	JES3 5.2.1
JES2 5.1 + PTF	JES3 5.1.1 + PTF
JES2 4.3 + PTFs	JES3 4.2.1 + PTFs
JES2 4.2 + PTFs	JES3 3.1.3 + PTFs
JES2 3.1.3 + PTFs	JES3 3.1.2 + PTFs

Notes:

- Details of PTF numbers required can be found in *MVS/ESA SP V5 Planning: Install and Migration JES2* and *MVS/ESA SP V5 Planning: Install and Migration JES3*.
- Please be aware that further detail on JES2 and JES3 implementation can be found in *MVS/ESA SP-JES2 Version 5 Implementation Guide* and *MVS/ESA SP-JES3 Version 5 Implementation Guide*.

2.2.2.1 JES2 and Sysplex

In this section we take a look at the implications of the JES2 level on any sysplex requirements:

- Single MVS image

  Customers are required to build a sysplex when MVS/ESA SP-JES2 Version 5 is installed. In a single system sysplex, XCF can be configured in local mode, so there is no requirement for a couple data set.

  If a JES2 release prior to MVS/ESA SP-JES2 Version 5 is used with MVS/ESA SP Version 5, a sysplex is not required.

- Multiple MVS images using one CPC

  Customers are required to use a sysplex when MVS/ESA SP-JES2 Version 5 is installed. XCF must be configured with a couple data set so that JES2 can keep track of member status.

  If a JES2 release prior to MVS/ESA SP-JES2 Version 5 is used with MVS/ESA SP Version 5, a sysplex is not required. For this type of configuration, a JES2 MAS must have seven or fewer MVS images.

  For customers using multiple MVS images, MVS/ESA SP-JES2 Version 5 cannot be mixed with other levels of JES2 in the same JES2 MAS.

- Multiple MVS images using multiple CPCs

  Customers are required to use a sysplex when MVS/ESA SP-JES2 Version 5 is installed. A Sysplex Timer is required to synchronize time across the CPCs.

  If a customer chooses to use a JES2 release prior to MVS/ESA SP-JES2 Version 5, a sysplex is not required. However, as with the single CPC configuration, the MAS can have no more than seven MVS images.

  You can run multiple systems in which some systems are running MVS/ESA SP 5.1, some are running MVS/ESA SP 5.2, and you are using a Version 5 MAS (multi-access spool). The MVS systems can be at any Version 5 release of MVS. If one of those systems is running MVS/ESA SP 5.1, you must run JES2 SP 5.1 on that system. In a MAS that has both JES2 SP 5.1 and JES2 SP 5.2 systems, you must install PTF UW14953 on the JES2 SP 5.1 systems.

  In an environment like the above, all the members of the JES2 MAS must be members of the same sysplex.

2.2.2.2 JES3 and Sysplex

In this section we take a look at the implications of the JES3 level on any sysplex requirements:

- Single MVS image

  Customers are required to build a sysplex when MVS/ESA SP-JES3 Version 5 is installed. In a single system sysplex, XCF can be configured in local mode, so there is no requirement for a couple data set.

  If a JES3 release prior to MVS/ESA SP-JES3 Version 5 is used with MVS/ESA SP Version 5, a sysplex is not required.

- Multiple MVS images using one CPC

  Customers are required to use a sysplex when MVS/ESA SP-JES3 Version 5 is installed. The couple data set required for XCF must be allocated and used.

  If a JES3 release prior to MVS/ESA SP-JES3 Version 5 is used with MVS/ESA SP Version 5, a sysplex is not required. For this type of configuration, the JES3 complex can have no more than eight members.

  For customers using multiple MVS images, MVS/ESA SP-JES3 Version 5.X.X cannot be mixed with other levels of JES3 in the same JES3 complex.

- Multiple MVS images using multiple CPCs

  Customers are required to use a sysplex when MVS/ESA SP-JES3 Version 5 is installed. A Sysplex Timer is required to synchronize time across the CPCs.

  If a customer chooses to use a JES3 release prior to MVS/ESA SP-JES3 Version 5, a sysplex is not required. However, as in the single CPC configuration, the JES3 complex can have no more than eight members.

2.2.2.3 Other JES3 Considerations

- JES3 consoles

  MVS/ESA SP-JES3 Version 5 Release 1.1 is the last release planned to support JES3 managed local consoles. Customers are encouraged to convert their JES3 consoles to other facilities, such as MVS and NetView consoles.

- JES3 tape utilities

  MVS/ESA SP-JES3 Version 5 Release 1.1 is the last release planned to support JES3 tape utilities. Customers are encouraged to convert their tape management to utilities supported by DFSMS.

2.3 Installation Requirements

In this section we take a look at what comprises MVS/ESA SP 5.2.0 and what system levels are required for the installation.

2.3.1 BCP Requirements

MVS/ESA SP 5.2.0 BCP consists of the following parts:

Table 2. MVS/ESA SP 5.1.0 FMIDs	
FMID	**Function**
HBB5520	Base Control Program (BCP)
JBB52N0	BCP US English
HCSH521	Hardware Configuration Dialogs (HCD)
JCSH523	HCD US English
HIO1104	IOCP
HTO1310	Base TSO

The installation of FMID HBB5520 supersedes all previous releases of MVS/ESA Version 3 and Version 4. Additionally, it deletes all previous versions and releases of MVS/ESA, MVS/XA and MVS/SP.

2.3.2 Driving System Requirements

Refer to *GC28-1428 MVS/ESA Planning: Installation and Migration with JES2 Program Directory* for further information.

2.3.3 Target System Requirements

Refer to *GC28-1428 MVS/ESA Planning: Installation and Migration with JES2 Program Directory* for further information.

2.3.4 I/O Configuration

An MVS/ESA SP 5.1.0 system can only be IPLed using an IODF. MVSCP definition is no longer supported.

Before the introduction of MVS/ESA SP V5, you could have used either an IODF or an MVSCP input data set to perform an IPL. To IPL MVS/ESA SP V5, you have to use an IODF.

When you have used a MVS/ESA SP V4.1 or V4.2 IODF, you have to upgrade your IODF before selecting any other option from the primary task selection panel. When you have used a MVS/ESA V4.3 IODF, you only have to upgrade the IODF when you want to change it.

When you have used MVSCP and IOCP input data sets so far, HCD provides a migration function via dialog or batch utility that lets you easily migrate your input data sets to an IODF. You can then use this IODF to IPL your MVS/ESA SP V5 operating system. Depending on your current operating system environment, there are different ways to migrate your operating system to MVS/ESA SP V5.

For more information about migration and installation, refer to *GC28-1428 MVS/ESA Planning: Installation and Migration with JES2* and *MVS/ESA SP V5 HCD: Planning*.

Further information on what is new in HCD in MVS/ESA SP 5.1.0 can be found in 4.6, “Hardware Configuration Definition Enhancements” on page 97.

2.3.5 Installation Recommendations

MVS/ESA SP 5.2.0 can be installed in the same way as previous versions or releases of MVS. These can be divided into two different approaches:

- System replacement method

  Examples of this method of installation are SystemPac/MVS or CBIPO.

- System upgrade method

  Examples of this method of installation are ProductPac/MVS or CBPDO.

For a complete discussion of these approaches, please refer to *MVS Software Management Cookbook* or *MVS Custom Built Offerings: Planning and Installation*.

2.4 Migration Considerations

This section describes the major migration considerations required for implementing MVS/ESA SP Version 5 in a non-sysplex environment. Additional considerations can be found in the following manuals:

- *MVS/ESA SP V5 Planning: Install and Migration JES2*
- *MVS/ESA SP V5 Planning: Install and Migration JES3*
- *MVS/ESA SP V5 Conversion Notebook*
- *MVS/ESA SP V5 JES2 Migration Notebook*
- *MVS/ESA SP V5 JES3 Conversion Notebook*
- *Sysplex Hardware and Software Migration*

2.4.1 I/O Definition Migration

This is potentially the most significant migration step required when installing MVS/ESA SP 5.2.0.

As mentioned previously in 2.3.4, “I/O Configuration” on page 24, MVS/ESA SP Version 5 HCD provides a batch migration utility for users migrating from MVS/SP Version 2 and Version 3.

Note: If you are migrating to MVS/ESA SP Version 5 from MVS/370, then you have to install MVS/ESA SP Version 5 using a SystemPac or the V5 CBIPO driver if you are planning to install via CBIPO.

If you are already an MVS/ESA SP Version 4 user, then you can use an IODF created using your current level of HCD to IPL a V5 system. You will, however, be required to upgrade that IODF when you use it with HCD 5.X. HCD prompts you when this is required.

Alternatively, if you are running MVS/ESA SP 4.2 or higher, you can install the HCD 5.X component on your current system and perform the migration in advance of installing Version 5 of MVS BCP.

The following list details the steps required to migrate your I/O definitions using the batch migration facility:

1. **Create a Work IODF**

   If an IODF does not exist into which you migrate your IOCP, MVSCP, or HCPRIO data sets, one has to be created.

2. **Migrate your input data sets**

   The HCD utility function for migration allows you to migrate the contents of your input data sets and store the definitions in the IODF.

3. **Analyze any errors and upgrade the input data sets**

   Any errors encountered during the migration process are reported in a HCD message log. These have to be reviewed, and the input source updated if necessary. Once the error has been fixed, the migration process can be rerun to build the work IODF. This process can be repeated as many times as necessary.

4. **Build a Production IODF**

   Once you have a “good” work IODF, you must convert this to a production IODF. You can only IPL the operating system using a production IODF.

This batch facility is fully described in *MVS/ESA SP V5 HCD: User's Guide*.

If your test environment is only a small subset of your total I/O configuration, we recommend only migrating those definitions necessary to IPL the test system using this approach. Once you have a MVS/ESA SP Version 5 system IPLed, you can then use the online HCD dialog to migrate the complete configuration.

2.4.2 Other Considerations

The majority of other migration tasks are dependant upon the environment in which you intend to use MVS/ESA SP 5.2.0. You should decide during planning which of the new functions or facilities you wish to make use of. One facility you have no choice over is 4-digit device support. An example of an area for consideration is automation. The device numbers in messages have 4-digit numbers (for example, device number 123 appears as 0123).

Note: If your automation routines are column sensitive, they will not work as expected.

A comprehensive list of items to consider can be found in *MVS/ESA SP V5 Planning: Install and Migration JES2* or *MVS/ESA SP V5 Planning: Install and Migration JES3*. These also give guidance on other planning considerations.

2.5 Changes to SYS1.PARMLIB

Have a brief look at what has changed in SYS1.PARMLIB. Full details of all the members and all of the options are described in *MVS/ESA SP V5 Initialization and Tuning Reference*. Additionally, the *MVS/ESA SP V5 Conversion Notebook* contains useful information on parmlib and other changes introduced with MVS/ESA SP Version 5.

Chapter 3. Exploitation of a Multisystem Environment

With the introduction of sysplex in MVS Version 4, the possibility of managing multiple systems coupled together by hardware elements became feasable. The initial restriction of having eight systems in a sysplex provided by MVS Version 4 has been removed by MVS/ESA SP Version 5 and it is now possible to couple up to 32 systems together in a sysplex. As more and more systems are added to a sysplex, so the complexity of managing them increases. This chapter discusses those functions in MVS/ESA SP Version 5 that address these complexities and eases the management tasks. These functions include:

- Coupling facility
- Cross-system extended services (XES)
- Sysplex failure management
- Enhanced event notification facility
- Serviceabilty enhancements
- Operator commands
- Operations services
- Sysplex device drain
- GRS enhancements
- Automation of IOS messages
- SMF enhancements
- Automatic restart management (ARM)
- MVS system logger

3.1 Coupling Facility

A coupling facility provides a shared storage with “intelligence.” A coupling facility is implemented by IBM Licensed Internal Code, known as coupling facility control code (CFCC). CFCC always runs in an LPAR in an ES/9000 511-based processor, an ES/9000 711-based processor, a S/390 9672, or a S/390 9674. A sysplex that includes a coupling facility and data sharing software is a parallel sysplex.

A sysplex in MVS/ESA SP Version 5 can exist without a coupling facility. However, a coupling facility is required for data sharing in a parallel sysplex. A sysplex can have more than one coupling facility. This is recommended for availability. The MVS systems that have paths to a coupling facility are not required to have CTC paths to each other, and CTCs can be an option, as shown in Figure 1. This greatly simplifies the complexity of defining the connections between the systems, and may eliminate the requirement for channels used only for XCF signalling.

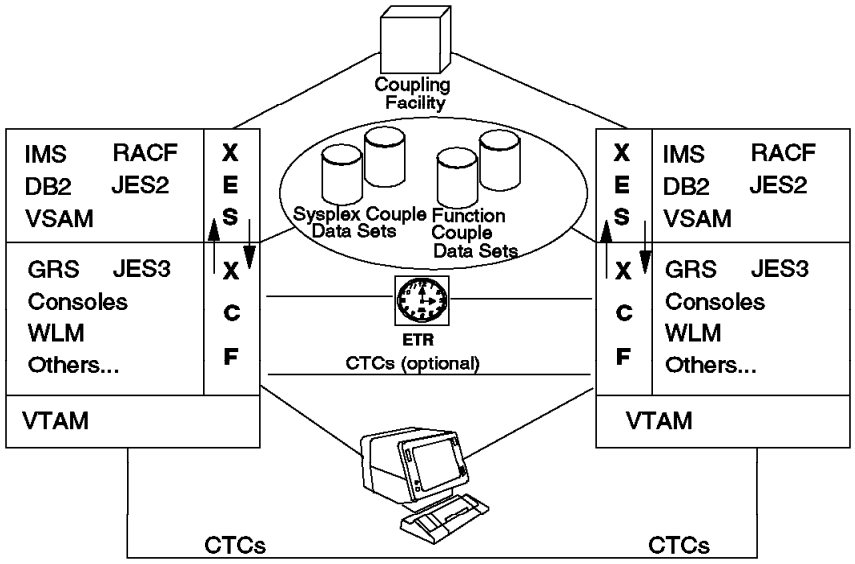


Figure 1. MVS/ESA Version 5 Sysplex Environment

The exploiters of XCF/XES with MVS/ESA SP Version 5 are:

- MVS workload manager
- JES2
- JES3
- VTAM
- CP/SM (CICSPlex system manager)
- AOC/MVS
- TSCF
- RMF
- RACF
- DFSMS/MVS
- IMS/ESA
- DB2 Version 4
- MVS system logger
- Automatic tape switching

3.2 Cross-System Extended Services (XES)

Cross-system extended services (XES) is an enhancement to cross-system coupling facility (XCF) introduced in MVS/ESA 4.1.0. XES provides the support for connection to a coupling facility. XES is a key to the growth from a small, single-system environment to a large multisystem environment. It provides facilities that allow for greater system availability and data sharing.

XES is a collection of services that enhance the XCF services introduced in MVS/ESA 4.1.0. XCF services provide the support for a sysplex in MVS. With XES, systems may be connected to a coupling facility.

To reduce complexity and cost in the management of the sysplex, XCF and XES provide a single-system image and improve the systems management characteristics of the sysplex environment.

XES provides the external MVS authorized programming interfaces and services that allow the subsystems and system components to gain access to coupling facility structures and to access and manipulate the data in each of these structures. XES defines three different structures and the services that manipulate these structures as follows:

Cache For a coupling facility cache structure, XES provides the interfaces for an authorized application or system component to define protocols that allow high-performance caching of user-defined data to be shared, with integrity, between occurrences of the application.

List For a coupling facility list structure, XES allows an authorized application or system component to define and manipulate lists (queues) of data items. High performance polling mechanisms can be implemented with this structure.

Lock For a coupling facility lock structure, XES provides shared and exclusive locking support to an authorized application or system component to allow for user-defined protocols and contention management in the high-performance serialization of logical resources as defined by the application.

Note: In a configuration without a coupling facility, attempts to use any of the structure services are rejected.

An installation defines the structures to be allocated in its coupling facilities by establishing a coupling facility resource management (CFRM) policy. The CFRM policy defines the name of each structure, the amount of coupling facility resource to be used for each structure, and to which coupling facility the structure should be assigned storage when it is allocated.

Note: An installation can predefine coupling facility policies in anticipation of adding one or more coupling facilities in the future, even though the configuration does not contain a coupling facility. The policies are stored in the CFRM couple data set.

3.2.1 Couple Data Sets

Systems in a sysplex wanting to use the coupling facility must have access to the coupling facility resource management (CFRM) couple data set. The other types of couple data sets are: the sysplex-wide shared sysplex, the sysplex failure management (SFM), the workload manager (WLM), the automatic restart management (ARM), and the MVS system logger (LOGR) couple data sets, as shown in Figure 2.

A couple data set is created through the XCF couple data set format utility and, depending on the designated type, is shared by some or all of the MVS systems in a sysplex.

The sysplex-wide shared sysplex couple data set can contain only data about the sysplex, systems, groups, and members that use the XCF services, but cannot store the CFRM, SFM, ARM, LOGR, or WLM policies.

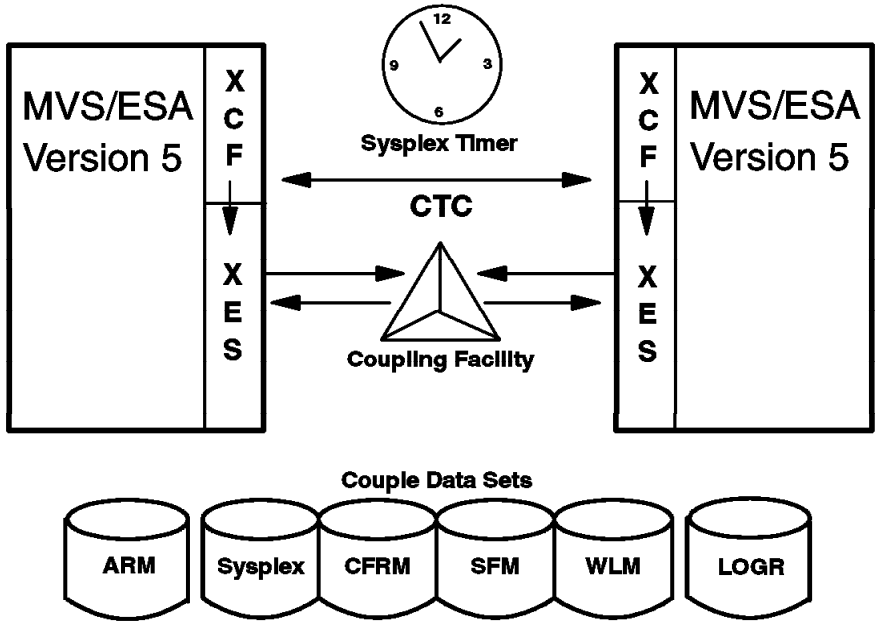


Figure 2. MVS/ESA Couple Data Sets

3.3 Sysplex Failure Management

MVS/ESA 4.1.0 introduced sysplex partitioning function. This function provides the ability to remove systems from a sysplex. In an MVS Version 4 sysplex, in the case of an MVS system failure, the other MVS systems in the sysplex can determine which system is to be removed, but cannot determine whether the system has been reset. The operator is responsible for resetting the failed system, and for confirming that the failed system is reset.

MVS/ESA SP Version 5 allows an installation to define which actions MVS takes when failures occur in a sysplex by defining a sysplex failure management (SFM) sysplex-wide policy. These actions are for the following failures:

- Signalling connectivity
- System failures when a system fails to update its status
- System failures in a PR/SM environment

Note: A coupling facility is required for SFM to handle signalling connectivity failures without operator intervention and when isolating system failures.

3.3.1 System Isolation

System isolation allows a failing system to be removed from the sysplex without operator intervention, and ensure data integrity of shared resources.

System isolation requires a coupling facility. On the failed system, system isolation terminates coupling facility accesses, resets channel paths and I/O, and loads a non-restartable wait state. The system that performs the sysplex isolation issues commands to the processor on which the failed system is running, routing the commands through the coupling facility. This process does *not* require any MVS function running on the failed system.

3.4 Enhanced Event Notification Facility

The event notification facility (ENF) provides a method for MVS to signal the occurrence of special events, each of which is assigned an event code. When one of these events occurs, ENF notifies all the users that have identified themselves as listeners for the event. The notification occurs by scheduling the execution of a routine established by the user code when it identified itself as a listener for an event.

The ENF enhancements in MVS/ESA SP Version 5 provide functions that can be used by coding parameters on the invocation of the ENFREQ LISTEN macro. The ENF enhancements introduced by MVS/ESA SP Version 5 are:

- Listener exit routines are allowed to run in the address space of the code that established the exit.
- The establisher of the listener exit can pass parameters to the listener exit routine.
- ENF can automatically delete the listener exit routine request at task termination or address space termination (EOT or EOM).

3.4.1 Additional Parameters for the ENFREQ Macro

With MVS/ESA SP 5.1.0, ENFREQ provides the following additional parameters:

- EOT
- EOM
- PARM
- SRBEXIT

A sample macro is shown as follows:

```
ENFREQ ACTION=LISTEN,    -- Function                          +
       CODE=ENFC35,      -- Event Code                        +
       SRBEXIT=(RO2),    -- Exit Address                      +
       PARM=LPARM,       -- Parameter                         +
       EOT=YES,          -- End-of-task delete indicator      +
       EOM=YES,          -- End-of-memory delete indicator    +
       ESTBNME=THISMOD,  -- Establisher Name                  +
       EXITNME=ENFLST01, -- Exit Name                         +
       DTOKEN=ENFL01T    -- Returned Token Field
```

3.4.2 Additional Event Codes for Signalling

Event codes have been added to allow ENF listeners to listen for the following events:

- Coupling facility availability
- MVS workload manager
- SRM
- ARM notification
- System Logger information
- DAE enhancements

To have a listener exit routine run in the address space of the code that established the exit, the ENFREQ LISTEN caller must specify the keyword *SRBEXIT*. When the listener exit routine is established by the SRBEXIT parameter, the exit routine runs under control of a local SRB scheduled in the address space of the code that established the exit. The use of the SRBEXIT improves usability, as the listener exit routine is able to reference the private storage of the code that established the exit; and improves reliability, as the routine runs isolated from the signalling program. If the listener exit routine is run under a local SRB, this also means that the routine can be loaded into private storage.

Note: SRBEXIT is not allowed for all event codes.

The parameter PARM on the invocation of the ENFREQ LISTEN macro allows a parameter list that is built by the caller to be passed to the listener exit routine. This simplifies communication between the code that established the listener exit and the listener exit.

ENFREQ LISTEN users are now able to specify what should be done in the event that the task or address space terminates. With the keywords EOT and EOM, the listener tells ENF that the listener request has to be deleted at task termination or address space termination, respectively, or that the listener request should survive the address space or task termination.

With the introduction of the coupling facility, the MVS workload manager, ARM support, the system logger, and modifications to DAE, some event codes have been added to let user programs be notified of important system-state changes. The additional event codes are:

Event Code 35 New coupling facility resources are available.

Event Code 38 An ARM notification event has occurred.

Event Code 41 An MVS workload manager event has occurred.

Event Code 42 A SRM event has occurred.

Event Code 43 A new copy of workload management sampled address space information is available via IWMRQRY.

Event Code 44 A coupling facility dynamic-related event has occurred.

Event Code 47 DAE now issues an ENF signal with this event code to communicate that the number of dumps scheduled for a given incident in the last interval has reached the threshold in ADYSETxx.

Event Code 48 Status information about log streams, the system logger component, and log stream coupling facility structures.

For a complete description of the parameters on the ENFREQ macro, and of the event codes, refer to *MVS/ESA SP V5 Authorized Assembler Services Guide* and to *MVS/ESA SP V5 Authorized Assembler Services Reference*.

3.5 Serviceability Enhancements

Enhancements to MVS/ESA SP Version 5 have been made in the area of production and management of dumps and traces. The enhancements help you to manage a sysplex or parallel sysplex environment with less effort and bring benefits in people productivity and disk space savings.

Dump enhancements in MVS/ESA SP Version 5 continue the evolution towards the goal of a single-system image that is easy to manage, no matter how large or complex. The following enhancements provide benefits for any size or complexity of an installation:

- Dump management
- SVC dump enhancements
- Stand-alone dump enhancements
- IPCS enhancements
- SLIP trap management
- Reducing wasted disk space for managing dumps

3.5.1 Dump Management Enhancements

Dump management is enhanced by modifications to the DUMPDS and DISPLAY DUMP operator commands. These help to control and display the options for automatic allocation of dump data sets.

Dump management has been improved to avoid incomplete SYS1.DUMPs due to the large amount of data to be dumped. This also avoids loss of dumps if all SYS1.DUMPs are full due to the number of dumps to be taken by all the systems in the sysplex.

Use of a multivolume data set is now possible for stand-alone dump to avoid loss of data if a SYS1.SADMP data set is too large to fit on a single volume. The operator can request additional areas to be dumped if the PROMPT option was selected at stand-alone program generation.

3.5.1.1 Automatic Allocation of Dump Data Sets

Dump management now provides the ability to automatically allocate dump data sets at the time the dump is created. The automatically allocated dump data sets are the exact size required to hold the dump, with no wasted space. The facility allows you to have the dump data written to a data set allocated either from a set of disk volumes or usinng an SMS storage class (if you have implemented SMS; refer to APAR OW03401 for enhancements in this support).

Note: This capability prefers SMS storage classes to disk volumes if you have both defined on your system. Allocation to SMS storage classes can use both multivolume data sets and striped data sets; allocations to disk can be neither of these.

After a dump has been captured, any free space is automatically released. For this reason, the recommended approach is to use dynamically allocated dump

data sets, even if you have some preallocated SYS1.DUMPxx data sets already in existence.

Automatic allocation is controlled using the DUMPDS operator command. The settings used in automatic allocation can be viewed with the DISPLAY DUMP operator command.

3.5.1.2 Automatic Allocation Failures

In the case that automatic allocation fails, the preallocated SYS1.DUMPxx data sets are used. If none are available for use, then the following message is issued:

```
IEA793A NO SVC DUMP DATA SETS AVAILABLE FOR DUMPID=dumpid FOR JOB
        (*MASTER*).
        USE THE DUMPDS COMMAND OR REPLY D TO DELETE THE CAPTURED DUMP
```

The dump remains captured in virtual storage until one of the following happens:

- Automatic allocation of a new dump data set by the operator to create a SYS1.DUMPxx using the DUMPDS ADD command.
- Writing to a preallocated SYS1.DUMPxx dump data set.
- The dump is deleted by the operator.
- The dump is deleted because the interval specified by the MSGTIME parameter on the CHNGDUMP command expires.

So, you can use preallocated dump data sets as a backup if the system is not able to allocate a data set automatically.

3.5.1.3 Commands for Automatic Allocation

The automatic allocation of dump data sets is controlled with the DUMPDS operator command. This command has two keywords added by MVS/ESA SP Version 5 as follows:

ALLOC Enables or disables automatic allocation.

NAME Establishes or modifies the name pattern to be used for automatically allocated dump data sets.

To enable automatic allocation of dump data sets, do the following:

- Define a dump data set naming convention.

  To control the naming convention used for data sets created by automatic allocation, use the command:

  ```
  DUMPDS ADD,NAME=name-pattern
  ```

  The default name pattern is

  ```
  SYS1.DUMP.D&DATE..T&TIME..&SYSNAME..S&SEQ.
  ```

  Names generated by your name pattern must follow the usual MVS data set naming conventions and limitations; otherwise they are rejected with the following message:

  ```
  IEE855I DUMPDS COMMAND RESPONSE
     DUMPDS COMMAND SYS1.DUMP DATA SET STATUS
     SYS1.DUMP DASD DATA SETS
     {ADDED: xx,xx,xx-xx,...| DELETED: uuu |CLEARED:}
     SYS1.DUMP DASD DATA SETS
     {NOT ADDED:|NOT DELETED:|NOT CLEARED:} text
  ```

- Define the automatic allocation resources.

  You do this by using the following command:

  ```
  DUMPDS ADD,VOL=volser
  DUMPDS ADD,VOL=(volser,volser ..)
  ```

  or

  ```
  DUMPDS ADD,SMS=class
  DUMPDS ADD,SMS=(class,class...)
  ```

- Decide whether to use automatic allocation.

  To enable or disable automatic allocation, you use the following:

  ```
  DUMPDS ALLOC=ACTIVE
  DUMPDS ALLOC=INACTIVE
  ```

  Note: The initial state of the system is with automatic allocation enabled, so you do not need to issue any command to enable the facility. If you disable the facility, no automatic allocation is done, naturally, and any subsequent dumps are written to the preallocated SYS1.DUMPxx data sets.

Table 3 shows the variables that can be used in the name pattern of the data set naming convention. The pattern can include text and variables; text is substituted in the variables when the data set name is generated.

Table 3. Variables for Use in Dump Data Set Names

Variable	Form	Length	Description
&SYSNAME	*sysname*	1 - 8	Name of system being dumped
&SJOBAME	*jobname*	1 - 8	Name of job requesting the dump
&DATE	*yymmdd*	6	Date dump was requested (equivalent to &YR2.&MON.&DAY.)
&YR4	*yyyy*	4	Year that dump was requested (4-digit year)
&YR2	*yy*	2	Year that dump was requested (2-digit year)
&MON	*mm*	2	Month that dump was requested
&JDAY	*ddd*	3	Day of the year that dump was requested
&DAY	*dd*	2	Day of the month that dump was requested
&WDAY	*ddd*	3	Day of the week that dump was requested (SUN, MON, TUE, WED, THU, FRI, SAT)
&TIME	*hhmmss*	6	Time dump was requested (equivalent to &HR.&MIN.&SEC.)
&HR	*hh*	2	Hour that dump was requested (24 hour clock)
&MIN	*mm*	2	Minute within hour that dump was requested
&SEC	*ss*	2	Second within minute that dump was requested
&SEQ	*nnnnn*	5	Sequence number for uniqueness

3.5.1.4 Automatic Allocation Commands

To authorize the automatic allocation of dump data sets and manage them, you can issue the commands shown in Figure 3. Placing the appropriate commands into the COMMNDxx parmlib member establishes the function at IPL.

If the DUMPDS NAME=..... command is not specified, the default name of the automatically allocated dump data sets has the following form:

```
SYS1.DUMP.D&date..T&time..&sysname..S&seq.
```

Some symbols such as *&date* for the date, *&mon* for the month, *&sysname* for the name of the system, or *&sysplex* for the name of the sysplex can be selected to customize the data set names. The *&seq* symbol is required to have a unique name for the automatically allocated data sets.

See *MVS/ESA SP V5 System Commands* for more information on the DUMPDS command, and *MVS/ESA SP V5 Diagnosis: Tools and Service Aids* for more information on SVC Dumps.

Note: Be careful with the data set name length as it cannot be longer than 44 characters.

The DUMPDS ADD,VOL=(*vol1*[,*vol2*]) command, shown in Figure 3, authorizes automatic allocation of the dump data sets on the non-SMS volumes specified.

The DUMPDS ADD,SMS=(*vol1*[,*vol2*]) command, shown in Figure 3, authorizes automatic allocation of the dump data sets on the STORAGE CLASSes specified instead of the volumes if SMS is active.

The DUMPDS ALLOC=ACTIVE command, shown in Figure 3, enables the automatic allocation of dump data sets.

```
DUMPDS NAME=SYS1.DUMP.D&DATE..T&TIME..&SYSNAME..S&SEQ
IEE855I DUMPDS COMMAND RESPONSE 542
DUMPDS COMMAND SYS1.DUMP DATA SET STATUS
  NAME PATTERN ACCEPTED: SYS1.DUMP.D&DATE..T&TIME..&SYSNAME..S&SEQ

DUMPDS ADD,VOL=TOTDL2
IEE712I CHNGDUMP PROCESSING COMPLETE
IEE855I DUMPDS COMMAND RESPONSE 325
DUMPDS COMMAND SYS1.DUMP DATA SET STATUS
  DASD VOLUMES ADDED: TOTDL2

DUMPDS ALLOC=ACTIVE
IEE712I CHNGDUMP PROCESSING COMPLETE
IEE855I DUMPDS COMMAND RESPONSE 327
DUMPDS COMMAND SYS1.DUMP DATA SET STATUS
  AUTOMATIC ALLOCATION IS: ACTIVE
```

Figure 3. Commands to Automatically Allocate Dump Data Sets

3.5.1.5 RACF Protection for Commands

If you use RACF or a similar security product, please take the following into consideration before activating automatic allocation of dump data sets:

- Create a user ID for DUMPSRV.
- Associate DUMPSRV′s user ID with the DUMPSRV started task.
- Authorize DUMPSRV′s user ID to create data sets using the naming convention you have defined.

3.5.1.6 DISPLAY Dump Command

To display the automatically allocated dump data sets, the DISPLAY DUMP command is enhanced. Figure 4 shows the parameter AUTODSN=nn. You can specify how many of the most recently automatically allocated dump data sets are to be displayed, up to 99. The value nn can also be specified as all. For more information on preallocated SYS1.DUMPxx data sets, and about automatically allocated dump data sets, see *MVS/ESA SP V5 Diagnosis: Tools and Service Aids*.

```
DISPLAY DUMP,TITLE,AUTODSN=5
IEE853I 13.59.47 SYS1.DUMP TITLES 328
SYS1.DUMP DATA SETS AVAILABLE=000 AND FULL=000
CAPTURED DUMPS=0000, SPACE USED=00000000M, SPACE FREE=00000500M
     SYS1.DUMP.D940608.T192346.LO6RS005.S00001 TITLE=ENF ABEND
            ERRORMOD=IEFENFNM
          DUMP TAKEN TIME=15.23.50 DATE=06/08/94
NO DUMP DATA AVAILABLE FOR 004 AUTOMATICALLY ALLOCATED DUMP DATA SETS
```

Figure 4. Displaying Automatically Allocated Dump Data Sets

3.5.1.7 Dump Analysis and Elimination

Throughout the discussion about the enhancements in dump processing provided by MVS/ESA SP 5.1.0, you should remember that the system occasionally requests dumps that you may not need. To keep from using your valuable system resources on unneeded dumps, you should suppress them with Dump Analysis and Elimination (DAE). This was introduced with MVS/ESA SP 4.3. For more information on DAE, see *MVS/ESA Version 5.1.0 Assembler Programming Guide*.

DAE is useful to manage the amount of SYS1.DUMPs taken by the system, eliminating some unneeded dumps. Systems in a sysplex can share the DAE data set. If a DAE data set is shared between multiple systems in a sysplex, DAE eliminates duplicate dumps across the whole sysplex.

In a sysplex environment, IBM recommends that systems share a single DAE data set. This reduces the number of SVC dumps and avoids or minimizes the risk of getting into an unavailable SYS1.DUMPxx situation. If intended dumps are missing, such as SLIP dumps or after a DUMP command, remember that DAE might have eliminated it.

With MVS/ESA SP 5.2.0, the DAE display facility provides a friendly interface to display and control entries in the DAE data set. It can be invoked by the ADYDSP TSO command or via IPCS option 3.5. Figure 5 shows the main panel.

```
--------------------------------- DAE Display --------------------------
                                                                Row 1 to 17
Command ===>                                                      Scroll ===
'SYS1.DAE' data set is assumed as no argument data set was given
  Enter "/" next to an entry to choose from a list of Action Codes.

  DATASET: 'SYS1.DAE'                              Total Dumps Requested
  Dumps since last Invocation 0                    Suppression Rate

   Last       Last      Event  Dump        SYMPTOM String information:
AC Date       System    Total  Taken       ABEND  REASON    MODULE    CSECT
-- ---------- --------  -----  ----------  -----  --------  --------  -----
   04/04/1995 SC49          6  04/04/1995  S00C4  00000010  IATLVVR
   04/03/1995 SC47         21  04/03/1995  S00C4  00000011  IEESB665  IEESB665
   04/03/1995 SC47         13  04/03/1995  S00C4  00000011  IGX00027  IGX00027
   04/03/1995 SC47          8  04/03/1995  S00C4  00000011  IGX00027  IGX00027
   04/03/1995 SC55         10  03/30/1995  S002D  D7D1F240  HASJES20  HASPCKPT
   04/03/1995 SC53         19  03/24/1995  S00C4  00000011  IEFW21SD  IEFTB726
   04/03/1995 SC49          6  03/19/1995  S0978  00000004  IATSSRE   IATSSRE
   04/03/1995 SC47          2  04/03/1995  S00C4  00000011  IKJEFLC   IKJEFLJU
   ...
```

Figure 5. DAE Display Facility Main Panel

The use of SUPRESSALL option in the ADYSETxx parmlib member is recommended. It will suppress all dumps that meet DAEs criteria for duplicate suppression. For improved suppression of rapidly occurring dumps that are duplicated, incoming dump requests are compared against captured dumps not yet written out to a dump data set. Only one dump is written for a given symptom string, and the information about it is communicated to any other systems in the sysplex that are sharing the same DAE data set. A simple procedure, using the DAE Display Facility, allows the installation to get a second dump, if needed.

Awareness of high-frequency dump requests is provided through event notification facility, event code 47, and controlled by the SVCDUMP option in the ADYSETxx parmlib member.

3.5.2 SVC Dump Enhancements

SVC dump is modified to support the creation of dumps using a data set naming convention that you control. In addition, SVC dump now automatically allocates its dump data sets, *of the correct size*, at the time the dump is written to disk storage. (SVC dump still supports the pre-MVS/ESA SP 5.1.0 structure of preallocated SYS1.DUMPxx data sets). This automatic allocation brings great benefits to:

- The MVS/ESA system programmer, who now does not need to determine:
  - How many dump data sets to allocate
  - What size to make them
  - What system to associate with which preallocated dump data set
  - What dump data sets need to be off-loaded and cleared
- The installation:
  - No need to preallocate dump data sets before you need them
  - No duplication of space associated with off-loading dump data sets

- No wasted space on disks associated with preallocated dump data sets that are larger than required
- No wasted space on disks associated with preallocated dump data sets that are too small, resulting in the need to recreate the problems to gather diagnostic data

3.5.2.1 SVC Dump Management

MVS/ESA SP 5.1.0 introduces enhancements to provide support for multisystem dumping. These enhancements enable you to gather the correct diagnostic information and data, no matter how complicated your installation, ranging from single systems up to a parallel sysplex environment. This represents an important aid to operator productivity in a complex installation. You need to make no changes to your existing environment to take advantage of the enhancements to SVC dump introduced by MVS/ESA SP 5.1.0. SVC dump management is enhanced in MVS/ESA SP 5.1.0 to:

- Enable one system in the sysplex to request a dump to be taken on one specific system or on several systems in the sysplex
- Support automatic data set allocation
- Provide an incident token to assist with problem management
- Obtain data from workload management and the coupling facility

There are two interfaces for requesting SVC dumps. The first is with the DUMP operator command. The second is by using one of two macros, SDUMPX and SDUMP. Both of these macros produce an unformatted SVC dump, but *IBM recommends using SDUMPX*, although MVS/ESA SP 5.1.0 still accepts the use of SDUMP. The SDUMPX macro has the following unique capabilities:

- Allows callers in access register (AR) mode to request an SVC dump (SDUMP only provides that service for callers in primary mode).
- Allows callers to include dataspaces in the SVC dump.
- Allows callers to include cross-system coupling facility (XCF), workload manager (WLM), and cross-system extended services (XES), and APPC information in the SVC dump.

For more information on the SDUMPX macro, see *MVS/ESA Programming: Authorized Assembler Services Reference, Volume 4 (SETFRR-WTOR)*.

In MVS/ESA SP 5.1.0, the DUMP command and the SDUMPX macro are improved to provide the following functions:

- Provide an external “incident token” for coordinating problem determination data
- Specify address spaces by name
- Dump information for XCF groups and members on remote systems
- Specify dataspaces by name
- Include problem description data in the dump
- Wildcard support in names for:
  - Systems
  - Address spaces
  - Dataspaces

- Group names
- Member names

SVC dump has also been modified to support automatic allocation of dump data sets, as described in 3.5.1.1, “Automatic Allocation of Dump Data Sets” on page 33.

3.5.2.2 Using Special Characters

You can specify wildcards in job names, dataspace names, user IDs, XCF group names, and XCF member names. The wildcards are as follows:

? Represents any character

***** Represents zero or more characters

The DUMP command or more precisely, the reply to the IEE094D message, has been improved to support a sysplex multisystem environment. For example, the reply to the IEE094D message could be entered as follows:

```
R xxx,REMOTE=(SYSLIST=(SC??('JES*')))
```

In the reply, all address spaces whose names begin with JES, and all systems whose names begin with SC followed by two and only two characters, are dumped.

Wildcards can start the string, end it, appear in the middle, or appear in several places in the string. A single ***** for the name indicates that all job names, dataspace names, user IDs, XCF group names, or XCF member names match. A single **?** indicates all names consisting of only one character. You can mix wildcards in any combination.

For more information on the DUMP command parameters, see *MVS/ESA SP V5 System Commands*.

3.5.2.3 Dump Options

Three keywords have been added in the dump options as follows:

REMOTE= Specifies the system or the systems in a sysplex you want to dump from another system in the sysplex.

STRLIST= Specifies a list of coupling facility structures names you want to be included in the dump.

PROBDESC= This keyword is intended for dumps requested by the REMOTE keyword. It provides problem information and specifies if a dump should be taken on another system in a sysplex.

The SDATA keyword has two added options:

WLM Request the workload management related data areas and storage to be included in the dump

XESDATA Request the coupling facility related information to be included in the dump

For a complete description of these parameters, see the DUMP command description in *MVS/ESA SP V5 System Commands*.

REMOTE keyword example: The REMOTE keyword can specify one or more systems by the XCF group and member names. You use the GRPLIST option to specify one or more systems by their names and, optionally, address spaces and job names to be dumped by those systems by using the SYSLIST option. Figure 6 shows the syntax of the REMOTE keyword.

```
REMOTE=(request[,request]...)]

Where request represents:

{GRPLIST={group(member)                                                  }  }
{        {(group(member[,member]...)[,group(member[,member]...)]...)     }  }
{                                                                           }
{SYSLIST={sysinfo|(sysinfo[,sysinfo]...)}                                   }
{       [,DSPNAME|,DSPNAME=(dspname-entry[,dspname-entry]...)  ]            }
{           [,SDATA|,SDATA=(option[,option]...)  ]                          }
{       [,STOR|,STOR=(beg,end[,beg,end]]                                    }
```

Figure 6. Syntax of REMOTE Keyword

The *sysinfo* option specifies the sysname and either the hexadecimal ASID identifier or the job name. The job name must be enclosed between quotes. It can contain wildcards. Figure 7 shows an example of the REMOTE parameter used to dump all jobs whose name begins with JES on all systems whose the name begins with SC. Refer to paragraph 3.5.2.2, “Using Special Characters” on page 40 for details about the special characters.

```
 DUMP COMM=(TEST2)

*374 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 R 374,REMOTE=(SYSLIST=(SC*('JES*')))
 IEE600I REPLY TO 374 IS;REMOTE=(SYSLIST=(SC*('JES*')))

 IEA794I SVC DUMP HAS CAPTURED: 489
 DUMPID=011 REQUESTED BY JOB (*MASTER*)
 DUMP TITLE=TEST2

 IEF196I IGD100I OFC6 ALLOCATED TO DDNAME SYS00011 DATACLAS (       )
 IEF196I IEF285I SYS1.DUMP.D940609.T201916.LO6RS005.S00010 CATALOGED
 IEF196I IEF285I VOL SER NOS= TOTDL2.
 IEA611I COMPLETE DUMP ON SYS1.DUMP.D940609.T201916.LO6RS005.S00010 497

 DUMPID=011 REQUESTED BY JOB (*MASTER*)
 FOR ASID (0001)
 REMOTE DUMPS REQUESTED
 INCIDENT TOKEN: WTSCPLX1 LO6RS005 06/09/1994 20:19:16
```

Figure 7. Example of the REMOTE Keyword of the Dump Command

STRLIST keyword example: Another enhancement in the reply to message IEE094D issued after the DUMP command is the STRLIST option to dump a list of coupling facility structures. Figure 8 shows the options of the STRLIST parameter.

```
STRLIST=(s-option[,s-option]...)]

where s-option represents:

STRNAME=strname
 [,CONNAME=conname]
 [,ACCESSTIME={ENFORCE|NOLIMIT|NOLIM}]
 [,LOCKENTRIES]
 [,USERCNTLS]
 [,({COCLASS|STGCLASS|LISTNUM}={ALL|(list)}         ]
 [    {[,ADJUNCT={CAPTURE|DIRECTIO}]           ]    ]
 [          [,ENTRYDATA={UNSERIALIZE|SERIALIZE}]    ]
 [    {[,SUMMARY]            })                     ]
```

Figure 8. Syntax of STRLIST Keyword

Following is a brief description of each of the structure-related keywords:

STRNAME= Designates a particular coupling facility list or cache structure by specifying the structure name.

CONNAME= Specifies the name of a connected user and is used only for a coupling facility cache structure.

ACCESSTIME= Indicates if the dump time limit specified on the ACCESSTIME parameter of the IXLCONN macro is in effect. When ACCESSTIME=ENFORCE is specified, the system holds the structure dump serialization no longer than the time interval specified on the IXLCONN macro. This is the default. If ACCESSTIME=0 is specified on the IXLCONN macro and ACCESSTIME=ENFORCE is specified on the dump request, the structure is not included in the dump. When ACCESSTIME=NOLIMIT is specified, the dump time limit is not in effect, and the system holds the structure dump serialization until processing is completed.

LOCKENTRIES The lock table entries for the requested coupling facility list structure are included in the dump. Lock table entries do not exist for a coupling facility cache structure.

USERCNTLS Requests that the user attach controls are included in the dump.

COCLASS= Specifies which cast-out classes are to be included in the dump. It is valid only for a coupling facility cache structure. Cast-out class controls for all cast-out classes are dumped if you specify ALL, or only cast-out class controls for the list of specified cast-out classes are dumped if you specify the list.

STGCLASS= Specifies which storage classes are included in the dump. It is valid only for a coupling facility cache structure.

LISTNUM= Specifies which lists are included in the dump. It is valid only for a coupling facility list structure.

ADJUNCT Indicates that the adjunct data for each entry specified by the range is included in the dump. When ADJUNCT=CAPTURE is specified, the adjunct data is captured in the facility dump space along with the directory information while dumping serialization is held. When ADJUNCT=DIRECTIO is specified,

the adjunct data is written directly to the dump data set after the directory information is captured.

ENTRYDATA= Indicates that the entry data for each entry within the requested range is included in the dump. When ENTRYDATA=SERIALIZE is specified, the entry data is dumped while serialization is held. When ENTRYDATA=UNSERIALIZE is specified, the entry data is dumped after the structure dump serialization is released.

SUMMARY Only a summary of the range of classes is dumped. SUMMARY may not be specified with ADJUNCT or ENTRYDATA.

Note: If a large amount of data is requested, the system may not be able to completely dump all the data. You can expect to successfully dump up to a maximum of 47 structures if you specify no more than six ranges. If the system cannot dump all the data you requested, it prioritizes the data according to your specifications on the command.

See *MVS/ESA SP V5 System Commands* for further details about ranges, structures, and the prioritization.

Figure 9 shows the utilization of STRLIST parameter to request that the dump includes:

- Structure control data, user attach control information, and entry directory information for every entry, grouped by storage class for the CACHESTRUCTURE
- Structure control data, all lock table entries and all list entry controls, grouped by list for the LISTSTRUCTURE

```
 DUMP COMM=(TEST4)

*321 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 R 321,STRLIST=(STRNAME=CACHESTRUCTURE,USERCNTLS,(STGCLASS=ALL),
               STRNAME=LISTSTRUCTURE,LOCKENTRIES,(LISTNUM=ALL))

 IEE600I REPLY TO 321 IS;
        STRLIST=(STRNAME=CACHESTRUCTURE,USERCNTLS,(STGCLASS=ALL),
                 STRNAME=LISTSTRUCTURE,LOCKENTRIES,(LISTNUM=ALL)),END


 IEA794I SVC DUMP HAS CAPTURED: 360
 DUMPID=006 REQUESTED BY JOB (*MASTER*)
 DUMP TITLE=TEST4

 IEF196I IGD100I 0FC6 ALLOCATED TO DDNAME SYS00006 DATACLAS (         )
 IEF196I IEF285I   SYS1.DUMP.D183322.L06RS005.S00005     CATALOGED
 IEF196I IEF285I   VOL SER NOS= TOTDL2.
 IEA611I COMPLETE DUMP ON SYS1.DUMP.D183322.L06RS005.S00005 364
 DUMPID=006 REQUESTED BY JOB (*MASTER*)
 FOR ASID (0001)

 REMOTE DUMPS REQUESTED
 INCIDENT TOKEN: WTSCPLX1 L06RS005 06/09/1994  18:33:22
```

Figure 9. STRLIST Keyword of the Dump Command

PROBDESC keyword: PROBDESC= is a keyword of the DUMP command. It provides problem information that is passed to any SVC dump, but is intended for dumps requested by the REMOTE parameter. When a system requests a dump on another system in the sysplex, the system being dumped calls an IEASDUMP.QUERY exit routine. See *MVS/ESA SP V5 Auth Assembler Services Guide* for further details about the IEASDUMP.QUERY exit routine.

This routine uses the information to determine if the system should be dumped and, if so, what storage areas should be added to the dump. The IEASDUMP.QUERY exit routine suppresses the requested dump only if PROBDESC specifies SYSDCOND.

The syntax of the PROBDESC keyword is:

PROBDESC=*key-spec*

The format of the *key-spec* is either of the following:

- *key*
- ((*key,data)*[,*(key,data)*]...)

Where:

data 1- to 16-characters of information to be used by the IEASDUMP.QUERY exit routine. *data* is not used for IBM supplied key names.

key A 1- to 8-character value that corresponds to the SDPD_KLD_KEY field in the IHASPDP mapping macro. IBM supplied values for *key* are SYSDCOND and SYSDLOCL.

Where:

SYSDCOND Suppresses a dump on another system in a sysplex if the other system does not have an IEASDUMP.QUERY routine or if no IEASDUMP.QUERY routine returns a code of 0.

SYSDLOCL Requests the following:

- Dumps of other systems in a sysplex
- An immediate dump of the local system on which you are entering the dump command
- A second, deferred dump of the local system if a SYSLIST or a GRPLIST option of the REMOTE parameter includes the local system

SDATA keyword example: The example in Figure 10 shows the use of the SDATA keyword to dump the workload management related data areas and storage, and the coupling facility related information. The COUPLE option already existed in MVS/ESA SP 4.3. There is no change to this option. Figure 10 on page 45 shows the utilization of the subparameters WLM and XESDATA.

```
 DUMP COMM=(TEST3)

*380 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 R 380,SDATA=(COUPLE,WLM,XESDATA)
 IEE600I REPLY TO 380 IS;SDATA=(COUPLE,WLM,XESDATA)

 IEA794I SVC DUMP HAS CAPTURED: 704
 DUMPID=012 REQUESTED BY JOB (*MASTER*)
 DUMP TITLE=TEST3

 IEF196I IGD100I OFC6 ALLOCATED TO DDNAME SYS00012 DATACLAS (         )
 IEF196I IEF285I   SYS1.DUMP.D940609.T204031.LO6RS005.S00011    CATALOGED
 IEF196I IEF285I   VOL SER NOS= TOTDL2.

 IEA611I COMPLETE DUMP ON SYS1.DUMP.D940609.T204031.LO6RS005.S00011 708
 DUMPID=012 REQUESTED BY JOB (*MASTER*)
 FOR ASID (0001)
 INCIDENT TOKEN: WTSCPLX1 LO6RS005 06/09/1994  20:40:3
```

Figure 10. WLM and XESDATA Parameters of the Dump Command

3.5.3 Stand-Alone Dump Enhancements

Because of the amount of data to be gathered, stand-alone dump contains some enhancements as follows:

- Block size has been optimized to improve dump efficiency.
- When a SYS1.DUMP data set is filled at dump time, the operator is prompted to give another device.
- Support of Extended Count Key Data (ECKD) architecture to improve performance by allowing nonsynchronous I/O.

3.5.3.1 SYS1.SADMP Allocation

The AMDSADDD REXX utility has been enhanced to optimize the SYS1.SADMP allocation. The data set allocated has the following characteristics:

- The data set name *must* be SYS1.SADMP if you define several SYS1.SADMPs on different DASD. Take care to not catalog them because the name is unique.
- The block size is:
  - 20800 for a 3380 or a 9345 DASD
  - 24960 for a 3390 DASD
- The LRECL is 4160.
- The RECFM is FBS.
- The DSORG is PS.
- The data set must consist of a single extent.

3.5.3.2 SYS1.SADMP Utilization

As in MVS/ESA SP 4.3, the SYS1.SADMP data set that receives data to be dumped *must* be allocated using the AMDSADDD REXX utility. In MVS/ESA SP 4.3, when a SYS1.SADMP was filled, the dump was truncated. In MVS/ESA SP 5.1.0, the operator is prompted with message AMD001A to specify another output device. The SADMP program can continue dumping to any DASD or tape device supported by stand-alone dump.

3.5.3.3 Stand-Alone Dump Considerations

Stand-alone dump processing can use a 3390 DASD previously defined with a block size of 20800 in MVS/ESA SP 4.3. However, the allocated space is not fully utilized. Now the AMDSADDD REXX utility allocates 3390 DASD devices using a block size of 24960.

Remember to use the AMDSADDD REXX utility with option CLEAR to reinitialize it after copying data on another data set if AMDSADMP was generated with option REUSEDS=NEVER. Otherwise, when the SYS1.SADMP data set is full, AMDSADMP issues the AMD093I message, and the operator is prompted with the AMD001A message for another output device. This could be another DASD or a tape device.

When you allocate and initialize the SYS1.SADMP data set using the AMDSADDD REXX utility, take care which volume you allocate it on. SYS1.SADMP *must not* be on the same volume as those containing:

- The IPL text of stand-alone dump
- Page and swap data sets

Note: Particular attention must be taken to avoid contention with other data sets on the same volume that could be allocated by other systems in or outside the sysplex. This could extend the time to complete the dump.

For a complete description of the dump options, see *MVS/ESA SP V5 Diagnosis: Tools and Service Aids*.

3.5.4 IPCS Enhancements

The following enhancements have been made to IPCS in MVS/ESA SP 5.1.0:

- Up to 100 traces concurrently open
- SYSTEM STATUS command for multisystem SVC dump
- 4-digit year formatting of TOD clock for multisystem SVC dump
- Component trace support
- Changes to COMCHECK subcommand
- Message data block formatter
- XES support

MVS/ESA SP 5.2.0 added the following enhancements:

- SYSTRACE subcommand replaces the VERBEXIT TRACE
- Better support for lists of data sets
- Support for extended sequential data sets
- Independent subcommands defaults for each logical screen
- Support for symbols with literal definitions
- Sysplex dump index data set
- ADDDUMP subcommand
- Improvements for other IPCS subcommands

IPCS has been modified to increase the number of traces it can concurrently manage, thereby greatly improving diagnostic capability. The IPCS MERGE subcommand accepts a sequence of CTRACE and GTFTRACE subcommands. Each of these subcommands can be directed to obtain its input from separate dump and trace data sets. In MVS/ESA SP Version 5, you can use MERGE to work concurrently with up to 100 traces.

Note: This feature requires that DFSMS/MVS is also installed on your system.

The SYSTEM STATUS report in IPCS has been changed to format the incident token in the SVC dump header record.

IPCS has been modified to provide a 4-digit year format of the time-of-day (TOD) clock that each system maintains. The load module affected is BLSUMTOD. The time stamp obtained from the TOD clock is passed to this load module, which returns a 26-character area containing, among other things, the 4-digit year.

You can use the IPCS CTRACE command to format the OPS component trace entries in either an SVC dump or in a trace data set.

The COMCHECK subcommand has been changed. The following parameters are no longer accepted on the subcommand:

- DATABLKS
- MCSINFO

An IPCS formatter is provided to format the Message Data Block (MDB). You use this formatter to analyze a dump or trace data set containing OPS component trace entries. The command to invoke this is:

```
CBFORMAT address.STRUCTURE(mdb)
```

An IPCS subcommand, XESDATA, provides reports and analysis routines for supporting the cross-system extended services (XES) component. XESDATA has full IPCS dialog and online help support. Another subcommand, STRDATA, was added to IPCS. This provides you with information in dumps about coupling facility structures. The STRDATA option and supporting panels are added to the IPCS ANALYSIS panel. This allows you to build an STRDATA request by specifying options on the panels. MVS/ESA SP 5.1.0 provides the macro, IXLZSTR, the coupling facility structure data access service macro. This enables programs running in the IPCS environment to access dumped coupling facility data. The macro returns the information to you in an area that you specify when you call the macro.

With MVS/ESA SP 5.2.0, the SYSTRACE subcommand provides all the functions previously supplied by the TRACE verb exit, plus selection of trace entries by time interval and by single CPU.

A list of generated dump data sets is created by each MVS/ESA SP 5.2.0 sysplex, and authorized IPCS users are able to access and edit it.

IPCS now supports extended sequential data sets in systems where DFSMS services are available. They are the recommended repository for dumps, as they have greater capacity and better performance.

Commands entered from a logical screen in which a dump other than the default dump is being viewed now can have their own set of subcommands defaults.

The LITERAL primary command and subcommand have been added to associate an IPCS symbol with any literal value.

SYS1.DUMPnn data sets and automatically allocated dump data sets generated by a MVS/ESA SP 5.2.0 are automatically enumerated in the system data set called the sysplex dump index. Its default data set name is SYS1.DDIR.

The ADDDUMP command adds a dump to the list of dumps in a sysplex dump index or personal dump directory, and briefly accesses it to create initial symbols.

SETDEF, EVALDEF, COMPARE, DROPDUMP, EVALUATE, STATUS and FIND subcommands have minor changes. Usage of IPCS dialog primary commands and dialog defaults option has been slightly improved. Subtle changes in panels are visible starting with the IPCS primary option menu.

In addition, the DUMPID displayed on messages IEA611I/IEA911E, IEA793A, IEA794I and IEA799I, will be placed in the dump header so it can be retrieved and viewed in the IPCS status worksheet.

3.5.5 SLIP Command Enhancements

Before MVS/ESA SP 5.1.0, defining and maintaining SLIP traps, particularly in a complicated multisystem environment, was a time-consuming task that was also prone to errors. MVS/ESA SP 5.1.0 addresses this problem providing support to allow you to:

- Provide multisystem SLIP traps
- Provide job name support
- Provide wildcard support
- Reduce size of SLIP traps
- Obtain XES information

MVS/ESA SP 5.2.0 adds more enhancements to provide the following support:

- Dynamic PER traps
- Indirect addressing for RANGE keyword
- SLIP traps in SVC dumps
- Match messages
- WAIT state information

3.5.5.1 Multisystem SLIP Traps

Enhancements to the SLIP command enable an installation to use a single SLIP command across multiple systems, instead of an individualized SLIP command for each system. To use this facility, it is now possible to specify job names in some parameters to identify dataspaces. As job names could be different on several systems, it is possible to use wildcards with job names. This greatly simplifies some SLIP commands.

MVS/ESA SP 5.1.0 adds a keyword IDGROUP=*idgroup*. You use this keyword to name a group of related SLIP traps on one or more systems when you attempt to diagnose a multisystem problem. You can choose a name for *idgroup*; this can be from one to 16 alphanumeric or national characters. It identifies the trap as a member of a group of traps with the same idgroup name. The group is set up when the SLIP traps are specified. When one SLIP trap matches, it can disable all the traps from the same idgroup even if they are on different systems in the sysplex.

Once any of the traps in an idgroup reaches its MATCHLIM or PRCNTLIM limits, that trap and all others in the idgroup are automatically disabled by SLIP processing, regardless of which system the trap is on.

Note: MATCHLIM=m specifies that the SLIP trap is to be disabled after m matches, where m is an integer from 1 to 65535. PRCNTLIM|PL=p specifies, for PER traps, a software limit for PER processing by indicating a maximum percentage of system time that can be devoted to processing caused by PER interruptions. See *MVS/ESA Version 5.1.0 System Commands* for a more detailed discussion of these parameters and the SLIP operator command.

Traps are disabled on a last in first out (LIFO) basis. SLIP also issues the message IEE451I when it disables related traps in an idgroup.

Note: This message is issued only *once* to the console of the system that initiates the idgroup disablement. It is also issued only *once* to the hardcopy logs of all systems in the sysplex on which the traps are disabled. The message descriptor code is 4, and the routing code is 10. See *MVS/ESA Version 5.1.0 System Messages Vol. 4 IEC-IFD* for more information.

If the operator deletes or disables a trap that is part of an idgroup, only that particular trap is affected; all the other traps of the idgroup are unaffected.

You can have multiple traps on the same MVS/ESA system with the same idgroup name. Multiple idgroups can also exist in the sysplex. You can display idgroups with the command:

```
DISPLAY SLIP ID=idgroup
```

Note: IBM recommends that in a sysplex environment you set up two parmlib members, one to enable your SLIP traps, and another to disable the traps. This should be compatible with your existing operating procedures for the management of SLIP traps.

SLIP processing invokes the MVS macro IEAINTKN to create a unique incident token to pass to SDUMPX. Whenever the SLIP action is to take a multisystem SVC dump, SLIP invokes IEAINTKN on the local system and passes the returned 32-character token to the remote systems in the sysplex.

3.5.5.2 Job Name Support

SLIP is enhanced to support a job name on the following parameters:

- DSPNAME=*jobname*.dspname
- ASIDSA=(*jobname*)
- DSSA=(*jobname*.dspname)
- For DATA, LIST, SUMLIST, TRATA support:
  - *jobname*.addr

Before MVS/ESA SP 5.1.0, it was necessary to specify the ASID associated with the dataspace. Now the dataspace name can be specified as ′jobname′.dspname:

Where:

jobname The job name associated with the address space using the dataspace. You can use wildcards in the job name. The job name must be enclosed between single quotes.

dspname A 1- to 8-character name associated with the dataspace at its creation.

3.5.5.3 Wildcard Support

It is now possible for the MVS/ESA system programmer to define shorter traps through the use of wildcard search arguments. The primary purpose of wildcard specification is to allow you to define one set of SLIP traps and then clone them to all systems in your sysplex. Two wildcards are available as follows:

? This special character replaces only one character at its place in the job name.

***** This special character replaces zero or more characters in the job name. For SLIP commands, *'*' must* be a suffix and *cannot* appear alone.

You can now use wildcard characters when specifying names for dataspaces and job names. These wildcards can be used in:

- Any job name in the ASIDSA, DATA, DSPNAME, DSSA, JOBNAME, JOBLIST, LIST, SUMLIST and TRDATA parameters
- The dataspace name of the DSPNAME parameter
- The user ID or the job name in the JOBNAME parameter

You can use these characters anywhere in the specification of the name of both *jobname* and *dspname*, for example:

```
DSPNAME=("IRLM*."*IXL)
```

DSPNAME= targets all job names that start with the characters "IRLM" and that own dataspaces whose names end with "IXL." Another example is:

```
JOBLIST=("??IRLM"),REMOTE=(SYSLIST=S1,S2,S3)
```

This identifies all jobs with 6-character names ending with "IRLM." The trap is set once, and the *REMOTE=* keyword defines the systems in the sysplex on which to define the trap.

If a program check occurs when IRLM is in control, the trap matches, and an SVC dump is taken (ACTION=SVCD) on all systems in the sysplex. Naturally, if you have any other jobs in the system that begin with the characters IRLM, they will have unnecessary dumps taken against them. So you should ensure that your naming convention for your jobs permits this type of cloning.

3.5.5.4 XES Support

Enhancements have also been included in SLIP processing to allow you to include cross-system extended services (XES) information in a SLIP-initiated dump. The operator now specifies the option on the SDATA parameter, as follows:

```
SLIP SET,...,SDATA=(XESDATA)...
```

The option follows all the syntax rules of the other SDATA options. The XESDATA option helps you to diagnose errors relating to XES, and includes the following data in a dump:

- XES component data
- Tracing information
- Locking information

3.5.5.5 Dumping the Coupling Facility Structures

For SLIP traps that specify ACTION=SVCD or ACTION=SYNCSVCD, the operator can specify the option, STRLIST. This option includes coupling facility structures in the dump. The dump includes only those structures that reside on coupling facilities that are connected to the system taking the dump.

You can include in a dump a list of coupling facilities. To do that, two keywords, STRLIST and STRNAME were created in MVS/ESA SP 5.1.0.

STRLIST is used to include a list of coupling facility structures (STRNAME). Its syntax is:

STRLIST= or STL=(STRNAME=*strname....*)

You may include more than one STRNAME=*strname* within parentheses separated by commas.

As the syntax of the STRNAME= parameter of the SLIP command is identical to the STRNAME= parameter of the DUMP command, please refer to “STRLIST keyword example” on page 41 for the description of this keyword.

3.5.5.6 OK Keyword

Another minor SLIP enhancement is the OK keyword. When you specify this keyword in the SLIP command, SLIP omits checking that could result in WTOR messages IEE604D or IEE831D. IBM recommends that you use this keyword *only* when issuing SLIP commands from parmlib. See *MVS/ESA SP V5 System Commands* for further details on the SLIP command.

3.5.5.7 MVS/ESA SP 5.2.0 SLIP Enhancements

The keyword TARGETID=*slipid* allows a PER trap, once it matches and reaches matchlim, to activate a second PER trap. ACTION=TARGETID must also be specified. There is no limit on the number of traps that can be chained this way.

Indirect addressing (register value, storage value) can now be used for the RANGE keyword, as it already was, for example, in the DATA keyword.

As several SLIP traps to produce dumps could be active at the same time, the EBCDIC text form of the SLIP command is now included in the dump, so the user will be sure on how this dump was produced.

The SLIP messages that indicate the trap has matched, will also include the jobname and ASID.

When SLIP puts the system in a restartable WAIT state as requested by A=WAIT keyword, the corresponding message IEE844W now provides the access registers.

The *IEA412I* message now includes the return and reason codes from SDUMP, helping the operator to find out why the system did not take a particular SVC dump.

3.5.6 Component Trace Enhancements

Component trace, in MVS/ESA SP 5.2.0, has been enhanced to provide:

- Better data integrity
- A more user friendly interface
- Improved options management relating to sub-level traces of a head trace
- Improved buffer management documentation

CTRACE, CTRACECS and CTRACEWR macros have been enhanced to identify situations when trace data is lost; messages are generated to warn the user of the occurance. In particular, CTRACE keeps an indication that some data has been lost if, at any time, the user′s trace data is not succeessfully written. This indicator is recorded when the next record is successfully written. If no subsequent record is written, message

```
ITT120I SOME COMPONENT TRACE DATA HAS BEEN LOST
```

is displayed when the external writer stops. The user should always check the return code from the CTRACEWR service.

The WRITER keyword of the CTRACE macro implements a simple mechanism to create external traces. It replaces the need to use WTRSTART, WTRSTOP, and WTR in a CTRACE parmlib member; it also replaces the need to issue the corresponding operator commands. The WRITER member name now applies to every sublevel trace specified.

The same CTncccxx parmlib member can now be used to both start and stop the trace. CTRACE does not require the ON and OFF specification in the parmlib member; the default is ON, and any other specification is ignored.

CTRACE is changed so that when a head level trace is turned off, all its sublevels traces are also be turned off, regardless of whether the sublevel was originally created with the LIKEHEAD keyword or not. This allows the user to truly turn the trace off.

CTRACE is also changed so that the connection to the writer will not be affected by a change in the sublevel trace options.

Buffer management is now clearly documented in the manuals to avoid integrity exposures.

3.6 Operator Commands

With MVS/ESA SP Version 5, changes have been made to operator commands and to consoles services, that, in MVS/ESA SP Version 5, is called operations services. In addition, changes to the WTOR and action messages have been implemented. The RMAX value used in message ID processing for Write To Operator with Reply (WTORs) can be changed by operator command; it also has a higher minimum value in a sysplex (99). Action message sequence numbers are changed in MVS/ESA SP Version 5.

Refer to the appropriate sections in this book and to *MVS/ESA SP V5 System Commands* for a more detailed description to the command changes.

The following operator commands are changed in MVS/ESA SP Version 5.

ROUTE The ROUTE operator command has been enhanced in MVS/ESA SP Version 5 so that the operator can use this command to send a command to all systems, or to a selected group of systems in a sysplex. This allows the operator to send a command to more than one system in a sysplex. Responses to this command are aggregated and returned to the operator on the originating system.

You can now route a command:

- To a *list* of specific systems
- To a *system group*

  For *system groups* definition refer to IEEGSYS parmlib member.
- Using *system symbols* for one or more systems in the sysplex

  Refer to 4.1, “Symbolic Substitution Overview” on page 77 and to *MVS/ESA SP V5 Initialization and Tuning Reference* for an explanation of *system symbols* and how to define them.

In MVS/ESA SP 5.2.0, the option *OTHER is added. It routes a command to all systems in the sysplex *except* the system on which the command is entered.

MODIFY The MODIFY command supports with MVS/ESA SP 5.1.0 wildcard characters in the identifier portion of the *procname.identifier* specification.

With MVS/ESA SP 5.2.0, you can now specify a trailing asterisk(*) on the *jobname* parameter value to apply the MODIFY command to more than one job or started task.

START The syntax has changed to support 4-digit device numbers. You must specify a slash(/) before the 4-digit device numbers.

MVS/ESA SP 5.2.0 introduces the JOBNAME and JOBACCT parameters for started tasks.

STOP Starting with MVS/ESA SP 5.1.0, the STOP command supports wildcard characters in the identifier portion of the *procname.identifier* specification.

SETPROG The dataset specified in the DSNAME parameter for SETPROG EXIT does not need to be APF-authorized in MVS/ESA SP 5.2.0.

SETXCF In conjunction with the parameters to enhance the support using a coupling facility, keywords are added in MVS/ESA SP 5.2.0 to the TYPE parameter. They are ARM and LOGR.

The ALTER keyword is added to SETXCF START/STOP processing in MVS/ESA SP 5.2.0.

CANCEL The CANCEL command supports wildcard characters with MVS/ESA SP 5.1.0 in the identifier portion of the *procname.identifier* specification.

The parameter ARMRESTART was added to the CANCEL command in MVS/ESA SP 5.2.0.

DISPLAY The DISPLAY ACTIVE command now accepts a *procname.identifier* specification. Additionally, wildcard characters can be specified in the identifier on a DISPLAY command.

Additional parameters were added in MVS/ESA SP 5.2.0:

- LOGREC displays the current medium as well as any alternate medium for it.
- SYMBOLS displays the static system symbols and associated substitution texts that are current.
- Several parameters were added for the SSI option.
- The output is changed for the following DISPLAY commands:
  - CONSOLES
  - XCF,STRUCTURES
  - M=STORAGE

FORCE The FORCE command with MVS/ESA SP 5.1.0 accepts a *procname.identifier* specification.

The parameter ARMRESTART was added in MVS/ESA SP 5.2.0, which allows the forced batch job or started task to be automatically restarted if registered as an element to the automatic restart manager.

VARY The VARY command has been restructured with MVS/ESA SP 5.1.0. The VARY CN command should be used for management of all MCS console functions.

The VARY OPERLOG,HARDCOPY command is introduced in MVS/ESA SP 5.2.0. This command is used to deactivate or activate the operations log.

3.6.1 ROUTE Command Changes

MVS/ESA SP Version 5 continues to build on the single-system image by providing an enhanced ROUTE operator command. The command sent using the ROUTE command is processed on the target system or systems. The additional parameters on the ROUTE command are as follows:

***ALL** Route the specified command to all systems in the sysplex.

system_group_name Route the specified command to the systems that comprise the specified system_group_name in the sysplex. To see how to specify a *system_group_name* refer to member IEEGSYS of SYS1.SAMPLIB.

T= The operator has the ability to specify an optional timeout value, T when issuing the ROUTE command. The default timeout value, T, is specified in the CONSOLxx parmlib member on the INIT statement.

***OTHER** Route the specified command to all systems in the sysplex, *except* the system on which the command was entered.

If you enter the following command,

```
ROUTE *OTHER,QUIESE
```

it is to be routed to all other systems in the sysplex excluding the sytem where you entered the command.

MVS/ESA SP 5.2.0 gives you the capability to specify a *list of systems, group names* or *sytem symbols* in ROUTE commands. If you want to create *system*

groups, refer to the IEEGSYS parmlib member. Figure 11 on page 55 shows an example of a ROUTE command using *sysnames*.

```
ROUTE (SC49,SC55),D A
 IEE421I RO (LIST),D A 509
 SC49     RESPONSES ------------------------------------------------
 IEE114I 15.20.28 95.073 ACTIVITY 282
  JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
 00001    00014    00000      00020    00019    00000/00030       00003
 SC55     RESPONSES ------------------------------------------------
 IEE114I 15.20.28 95.073 ACTIVITY 469
  JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
 00002    00018    00003      00020    00010    00003/00030       00003
```

Figure 11. Example of ROUTE Command Using Sysnames

3.6.1.1 System Symbols Used with the ROUTE Command

With MVS/ESA SP 5.2.0, you can specify *system symbols* in commands that are routed with the ROUTE command to one or more systems in a sysplex. To find out what are the symbol definitions, you can enter the DISPLAY SYMBOLS command; it will display the current static *system symbols* and their associated substitution texts.

The following example starts a job on each system in the sysplex as a started task. The JCL was added to the SYS1.STCJOBS data set, shared by all systems in the sysplex. On each system, the job gets another jobname, depending on the system symbol *&sysname*. Figure 12 shows an example of a ROUTE command using *system symbol*.

```
RO *ALL,S TESTSYM,JOBNAME=J&SYSNAME
   RO T=005,SC54,S TESTSYM,JOBNAME=J&SYSNAME
   RO T=005,SC52,S TESTSYM,JOBNAME=J&SYSNAME
   RO T=005,SC50,S TESTSYM,JOBNAME=J&SYSNAME
   RO T=005,SC49,S TESTSYM,JOBNAME=J&SYSNAME
   S TESTSYM,JOBNAME=J&SYSNAME
   RO T=005,SC53,S TESTSYM,JOBNAME=J&SYSNAME
   IEE295I COMMAND CHANGED BY SYMBOLIC SUBSTITUTION 719
   ORIGINAL: S TESTSYM,JOBNAME=J&SYSNAME
   MODIFIED: S TESTSYM,JOBNAME=JSC52
   S TESTSYM,JOBNAME=J&SYSNAME
   S TESTSYM,JOBNAME=J&SYSNAME
   S TESTSYM,JOBNAME=J&SYSNAME
   IEE295I COMMAND CHANGED BY SYMBOLIC SUBSTITUTION 929
   ORIGINAL: S TESTSYM,JOBNAME=J&SYSNAME
   MODIFIED: S TESTSYM,JOBNAME=JSC54
   ......
   ......
```

Figure 12. ROUTE Command Using System Symbols

Figure 13 shows the started task running using system-ID SC50 as an example.

```
RO SC50,D A,J&SYSNAME
  D A,J&SYSNAME
  IEE295I COMMAND CHANGED BY SYMBOLIC SUBSTITUTION 276
  ORIGINAL: D A,J&SYSNAME
  MODIFIED: D A,JSC50
  D A,JSC50
  IEE115I 11.25.36 95.075 ACTIVITY 278
   JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
  00001    00015    00001      00020    00019    00001/00030       00002
   JSC50    WTEST    STEP1    OWT  S   A=0019   PER=NO   SMC=000
                                       PGN=N/A  DMN=N/A  AFF=NONE
                                       CT=000.252S  ET=00.17.48
                                       WUID=JOB19063 USERID=STC
                                       WKL=SYSTEM   SCL=SYSSTC   P=1
                                       RGP=N/A      SRVR=NO  QSC=NO
```

Figure 13. Example for ROUTE DISPLAY Command Using System Symbols

You may cancel these jobs by using the following command:

```
RO *ALL,C J&SYSNAME..*
```

The first dot after *sysname* is for the end of symbol; the second is for the next jobname qualifier.

3.6.1.2 CONSOLxx Changes for the ROUTE Command

The ROUTTIME parameter, defined in the CONSOLxx parmlib member, specifies the amount of time the sending system waits for command responses from each system before aggregating the responses. It is defined as follows:

```
INIT     ROUTTIME(nnn)
```

The value nnn can range from 0 to 999 seconds and specifies how long the sending MVS system waits for responses from each of the target systems before aggregating the responses and displaying the results. If nnn is 0 then no aggregation is done, and the responses from each system are sent individually to the operator.

This value can be changed by the operator; for example, to change the route time to 60 seconds for *one* command, you add T=60 to the command:

```
ROUTE T=60
```

To override the timeout value specified in CONSOLxx for all systems in the sysplex, and set it to 60 seconds, you can issue the following command:

```
K M,ROUTTIME=60
```

The timeout value can be displayed by the operator using the command:

```
K M
```

Note: The enhancements to the ROUTE command are rejected by those systems in a sysplex running versions of MVS/ESA earlier than MVS/ESA SP 5.1.0. The aggregated command response indicates “no response received” from the down-level systems.

IBM recommends that you do not use ROUTE to issue commands that result in the following responses:

- A WTOR
- A command response without a console ID

- An excessive amount of data (for example, a display of all units)

3.6.2 MODIFY/STOP/CANCEL Command Changes

Starting with MVS/ESA SP 5.1.0, the MODIFY, STOP and CANCEL operator commands support wildcard search arguments in the identifier portion of the command. This allows a single issuance of the command to act on more than one started task by accepting wildcards in the identifier field that follows the procedure name.

Previously, the format of the command was:

```
F proc.id,text
```

The ability to specify wildcard search arguments in the identifier portion of the command allows the following possibilities:

```
F proc.*,text
F proc.id*,text
```

Note: STOP and CANCEL are both changed to accept wildcard searches in the same way as the MODIFY examples shown.

MVS/ESA SP 5.2.0 allows you to specify a trailing asterisk (*) on the *jobname* parameter to apply the MODIFY command to more than one job or started task:

```
F *.YZ,parameters
```

For more information on how to use asterisks with a MODIFY command, refer to *MVS/ESA SP V5 System Commands*.

3.6.3 DISPLAY/FORCE Command Changes

Both the DISPLAY and FORCE operator commands have been enhanced to support a procname.identifier specification when displaying information about an active started task and when terminating an active started task.

The DISPLAY command accepts a wildcard search argument in the identifier portion of the command. The identifier can be an asterisk, or one or more characters followed by an asterisk. The asterisk format means to retrieve information about all active started tasks that begin with the specified characters.

```
D A proc.*
F proc.id*
```

In MVS/ESA SP 5.2.0, the following parameters are added to the DISPLAY command:

- LOGREC displays the current Logrec recording medium as well as any alternate available medium. It indicates whether the log is been recorded in a data set or in a logstream.
- SYMBOLS displays the static system symbols and the associated substitution text that are currently in effect on that one system.
- UNITS,,AUTOSWITCH|AS displays units in autoswitch status
- SSI option allows the operator to display information about subsystems.
- The TYPE parameter of the DISPLAY XCF,COUPLE command has two additional options to display couple data set information:

ARM for the automatic restart manager

LOGR for the system logger

Figure 14 shows an example of a display LOGREC command and its output.

```
D LOGREC,ALL
IFB090I   15.50.25  LOGREC DISPLAY 372
 CURRENT MEDIUM = DATASET
    MEDIUM NAME = SYS1.SC54.LOGREC
 DATASET MEDIUM = SYS1.SC54.LOGREC
```

Figure 14. Display LOGREC Information - Data Set

Figure 15 shows an example of a display LOGREC command after switching to the *logstream* via the SETLOGRC command.

```
D LOGREC,ALL
IFB090I   10.19.25  LOGREC DISPLAY 017
 CURRENT MEDIUM = LOGSTREAM
    MEDIUM NAME = SYSPLEX.LOGREC.ALLRECS
         STATUS = CONNECTED
 DATASET MEDIUM = SYS1.SC54.LOGREC
```

Figure 15. Display LOGREC Information - Logstream

Figure 16 shows an example of a DISPLAY SYMBOLS command.

```
D SYMBOLS
IEA007I STATIC SYSTEM SYMBOL VALUES 609
        &SYSCLONE. = 54
        &SYSNAME.  = SC54
        &SYSPLEX.  = WTSCPLX1
```

Figure 16. Display Symbols Command

Figure 17 shows an example of a DISPLAY UNIT,,AS command.

```
D U,,AS,,1
IEE343I 18.51.09 UNIT STATUS 808
UNIT  TYPE  STATUS     SYSTEM    JOBNAME   ASID  VOLSER   VOLSTATE
0B3F  349S                                                 /REMOV
```

Figure 17. Display Unit Command - Autoswitch

Note: If the volume is allocated to another system, the STATUS, SYSTEM, JOBNAME, ASID, VOLSER and VOLSTAT information is retrieved from the coupling facility structure IEFAUTOS.

Figure 18 shown an example of the display SSI command.

```
D SSI,ALL
   IEFJ100I  13.42.20  SSI DISPLAY 124
   SUBSYS=JES2 (PRIMARY)
      DYNAMIC=NO      STATUS=ACTIVE       COMMANDS=N/A
    FUNC=  1   2   3   4   5   6   7   8   9  10  11  12  13  16  17  18
           19  20  21  53  54  64  70  71  75  77
   SUBSYS=MSTR
      DYNAMIC=NO      STATUS=ACTIVE       COMMANDS=N/A
    FUNC=  4   5   6   8   9  10  12  14  15  32  33  48  50  54  63  68
           72  73  78
   SUBSYS=SMS
      DYNAMIC=NO      STATUS=ACTIVE       COMMANDS=N/A
    FUNC=  8  15  55
```

Figure 18. Display SSI Command

For more information about the SSI commands see section 3.6.5, “SETSSI Command” on page 60. Figure 19 is an example of the DISPLAY command showing the couple data sets, pimary and alternate, used for the system logger. Also displayed are the systems using the logger facility, and systems that are not using the logger facility.

```
D XCF,COUPLE,TYPE=LOGR
 IXC358I  13.34.37  DISPLAY XCF 520
 LOGR COUPLE DATA SETS
 PRIMARY    DSN: SYS1.XCF.LOGR10
            VOLSER: TOTCAT     DEVN: 0FC7
            FORMAT TOD         MAXSYSTEM
            02/14/95 15:07:04         16
 ALTERNATE  DSN: SYS1.XCF.LOGR20
            VOLSER: TOTTS1     DEVN: 0FCC
            FORMAT TOD         MAXSYSTEM
            02/14/95 15:07:10         16
 SYSTEMS USING LOGR:
   SC49 SC50 SC52 SC54 SC55
 SYSTEMS NOT USING LOGR:
   SC47 SC53
```

Figure 19. Display System Logger Couple Data Sets

3.6.4 SETLOGRC Command

The MVS system logger is a set of services that allows an application to write, browse, and delete log data. You can use system logger services to merge data from multiple instances of an application, including merging data from different systems across a sysplex.

MVS writes records describing error, error related conditions and exception conditions for hardware and software products to a system data set named SYS1.LOGREC. In a multisystem complex or sysplex, each system (image) has its own LOGREC data set.

These records are copied by EREP, IBM′s error record report generator program, to an EREP history data set, the LOGREC data set is cleared, and system and product level reports are produced from one or more of the EREP history data sets.

If an installation has a sysplex, the installation can eliminate the effort required to define, manage and process records from multiple LOGREC data sets. By using a sysplex-wide LOGREC log stream instead of individual LOGREC data sets, the system can write records to and read records from the logstream. EREP, or any other program that processes the LOGREC records, can read the LOGREC log stream as though it were an EREP history data set.

The SETLOGRC command is added by this function.

```
D LOGREC,ALL
  IFB090I   15.31.42  LOGREC DISPLAY 401
   CURRENT MEDIUM = LOGSTREAM
      MEDIUM NAME = SYSPLEX.LOGREC.ALLRECS
    STATUS = CONNECTED
   DATASET MEDIUM = SYS1.SC54.LOGREC
   SETLOGRC DATASET
  IFB097I LOGREC RECORDING MEDIUM CHANGED FROM LOGSTREAM TO DATASET
   D LOGREC,ALL
  IFB090I   15.35.03  LOGREC DISPLAY 405
   CURRENT MEDIUM = DATASET
      MEDIUM NAME = SYS1.SC54.LOGREC
   DATASET MEDIUM = SYS1.SC54.LOGREC
   SETLOGRC LOGSTREAM
  IFB097I LOGREC RECORDING MEDIUM CHANGED FROM DATASET   TO LOGSTREAM 40
  IEF196I IXG201I REQUEST TO CONNECT TO LOGSTREAM SYSPLEX.LOGREC.ALLRECS
  IEF196I IN STRUCTURE SYSTEM_LOGSTREAM ACCEPTED.
  IXG201I REQUEST TO CONNECT TO LOGSTREAM SYSPLEX.LOGREC.ALLRECS 410
  IN STRUCTURE SYSTEM_LOGSTREAM ACCEPTED.
```

Figure 20. SETLOGRC Command

3.6.5 SETSSI Command

The SETSSI operator command allows the operator to dynamically add, delete, activate or deactivate a subsystem without the need to re-IPL the system. The subsystem itself determines whether the SETSSI command can be used to dynamically modify the subsystem. If you enter a SETSSI command for a subsystem that does not allow the SETSSI command, the system ignores the command and issues an error message.

The SETSSI command must be essued form a console with a MASTER-level authority. For more information on the dynamic SSI refer to 4.2.1, “Dynamic SSI” on page 86.

Figure 21 shows an example of adding and displaying a subsystem.

```
SETSSI ADD,SUB=TSYS,INITRTN=TSYSINIT
IEF196I TSYS - SUBSYSTEM INITIALIZED
TSYS - SUBSYSTEM INITIALIZED
IEFJ022I SETSSI ADD COMMAND FOR SUBSYSTEM TSYS COMPLETED 093
SUCCESSFULLY
D SSI,SUB=TSYS
IEFJ100I  11.04.40  SSI DISPLAY 217
SUBSYS=TSYS
   DYNAMIC=YES     STATUS=ACTIVE       COMMANDS=ACCEPT
```

Figure 21. Adding and Displaying a Subsystem

Figure 22 shows an example of deactivating a subsystem.

```
SETSSI DEACTIVATE,SUBNAME=TSYS
IEFJ022I SETSSI DEACTIVATE COMMAND FOR SUBSYSTEM TSYS COMPLETED 222
SUCCESSFULLY
D SSI,SUB=TSYS
IEFJ100I  11.08.00  SSI DISPLAY 224
SUBSYS=TSYS
   DYNAMIC=YES     STATUS=INACTIVE     COMMANDS=ACCEPT
```

Figure 22. Deactivating a Subsystem

3.6.6 VARY Command Changes

MVS/ESA SP 5.2.0 and MVS/ESA SP 5.1.0 introduce several changes and enhancemets to the MVS VARY command. These changes and enhancements are described in the following sections.

3.6.6.1 VARY OPERLOG Command

The operations log is a log stream that uses the system logger to record and merge communications about programs and system functions from each system in the sysplex. The OPERLOG provides a sysplex-wide merged and chronologically ordered message log.

The operations log is operationally independent of the system log. An installation can choose to run either or both of the logs. If you choose to run operations log as a replacement for the SYSLOG, you can prevent the future use of the SYSLOG. Once an operations log is started with the SYSLOG not active, enter the WRITELOG CLOSE command. You can set up the operations log to receive records from the entire sysplex or only from a subset of the systems, depending on the needs of the installation. You can dynamically activate or deactivate the operations log with the following command:

```
V OPERLOG,HARDCPY,OFF
  IEE338I OPERLOG  INACTIVE AS HARDCPY
  IEA631I  OPERATOR *OPLOG06 NOW INACTIVE, SYSTEM=SC54    , LU=NONE
   D C,HARDCOPY
  IEE889I 13.01.20 CONSOLE DISPLAY 560
  MSG: CURR=6    LIM=1500 RPLY:CURR=6    LIM=999  SYS=SC54      PFK=00
   CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
   SYSLOG                 COND=H      AUTH=CMDS         NBUF=0    UD=Y
                          ROUTCDE=ALL
   V OPERLOG,HARDCPY
  IEA630I  OPERATOR *OPLOG06 NOW ACTIVE,   SYSTEM=SC54    , LU=NONE
   D C,HC,L=USER1
  IEE889I 13.01.28 CONSOLE DISPLAY 564
  MSG: CURR=7    LIM=1500 RPLY:CURR=6    LIM=999  SYS=SC54      PFK=00
   CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
   SYSLOG                 COND=H      AUTH=CMDS         NBUF=0    UD=Y
                          ROUTCDE=ALL
   OPERLOG                COND=H      AUTH=CMDS         NBUF=N/A  UD=Y
                          ROUTCDE=ALL
```

Figure 23. Deactivation and Activation of OPERLOG

For more information about sysplex implementation see *MVS/ESA Version 5 Sysplex Implementation Guide*.

3.6.6.2 VARY dddd,AUTOSWITCH Command

MVS/ESA SP 5.2.0 introduces support for shared tapes in a sysplex environment. The VARY command can be used to dynamically change the autoswitch status of eligible devices. The AUTOSWITCH|AS keyword can only be used with offline devices. Figure 24 shows an example of the command.

```
VARY B3F,OFFLINE
IEF281I OB3F NOW OFFLINE
VARY B3F,AS,ON
IEE463I UNIT OB3F IS NOW DEFINED AS AUTOSWITCH CAPABLE.
VARY B3F,ONLINE
IEE302I OB3F     ONLINE
```

Figure 24. Vary Device Autoswitch

3.6.6.3 Console Support Changes

MVS/ESA SP 5.1.0 makes changes to the VARY operator command in order to simplify the task of managing MCS consoles. The changes affect the following commands and definitions:

- CONSOLxx definition
- Console name considerations
- VARY CN
- VARY ONLINE
- VARY OFFLINE
- VARY CONSOLE
- DISPLAY CONSOLES command output changed

CONSOLExx Definition Changes: The following changes are made to the CONSOLxx parmlib member:

NAME This keyword is now required and allows you to specify a name that uniquely identifies a console. This name can be used on the VARY command. Console names are specified for MCS consoles in the CONSOLxx parmlib member as follows:

```
CONSOLE  DEVNUM(devnum)
         NAME(cnname)
         SYSTEM(sysname)
```

The NAME(cnname) parameter defines the console name. If you do not specify NAME, the default is the name of the system to which the system console is physically attached. If the system name has already been taken, the default name is

```
SYSCNxxx
```

where *xxx* is a unique system-generated identifier.

If you specify a console name that is incorrect (invalid syntax or a duplicate name already in use by an extended MCS console), MVS/ESA SP 5.1.0 rejects the CONSOLE statement. This is a change from the way MVS/ESA Version 4 worked.

If the system console name duplicates the name of an existing console, the system assigns the default name to the system console. If you try to activate a console whose name is the same as an existing system console, the system rejects the activation.

SYSTEM An optional SYSTEM keyword is added and allows you to specify a system where the console should be expected to be active.

Note: The SYSTEM keyword should be used with extreme caution. If you have a sysplex with more than one CONSOLxx member, the first SYSTEM value associated with the console name is used. If more than one SYSTEM value occurs for a console name in the sysplex, then the system on which the console is activated depends upon the order in which the systems join the sysplex.

See *MVS/ESA Version 5 Initialization and Tuning Reference* for a complete description of the CONSOLxx parmlib member.

Console Name Considerations: For the sysplex environment, console names should be used because of the complexity of the operational setup. Most sysplex installations use a common CONSOLxx sysplex-wide parmlib member for console definitions to reduce the amount of administrative overhead in maintaining multiple versions.

If you currently run a single MVS/ESA image, you are still encouraged to implement console names. The use of console names positions you for growth into more complex MVS/ESA environments, including sysplex and parallel sysplex. If you use console names, you can use alternate console groups to provide greater flexibility and control in recovering from console failures. See *MVS/ESA SP V5 Planning: Operations* and *MVS/ESA SP V5 Initialization and Tuning Reference* for more information on setting up alternate console groups.

For new MVS/ESA customers, the mandatory use of console names is a positive step towards establishing effective operational procedures.

VARY CN Command: The VARY CN(console_name) command should be used to perform all MCS console management functions. This command enables the operator to manage an MCS console in a sysplex without needing to know to which system the console is attached. Additionally, the operator does need to know whether or not the console is active. The VARY CN command is sysplex-wide in scope.

The VARY CN(console_name) operator command activates, deactivates and sets attributes for MCS consoles. It also sets attributes for extended MCS consoles. You can specify console names on the VARY CN command.

Note: The VARY CN(console_name),ONLINE command does not accept extended MCS console names as input; you can only specify MCS console names.

MVS/ESA SP 5.1.0 introduces an enhancement that allows specifying an asterisk (*) as the name. The asterisk means the console from which you are currently issuing commands. You should use this command to activate a console instead of:

```
VARY xxx,CONSOLE
```

The following command should be used to deactivate a console:

```
VARY CN(console_name),OFFLINE
```

You can still use the VARY xxx,OFFLINE command in MVS/ESA SP 5.1.0 to manage the deactivation of MCS consoles, but IBM recommends migration to the

VARY CN(console_name),OFFLINE command, as the older form will no longer be enhanced.

The VARY CN(console_name),OFFLINE command deactivates the specified console regardless of where the device is attached in the sysplex. This means that the operator can issue the command from any system in the sysplex. Because of this, the operator may receive responses to the command from one or more members of the sysplex.

System Determination Steps: When you issue the VARY CN(console_name),ONLINE command, the following steps determine on which MVS/ESA system the console is activated:

1. The system that is defined on the SYSTEM=sysname keyword on the VARY command:

   ```
   VARY CN(cnname),ONLINE,SYSTEM=sysname
   ```

2. The system on which this console was last active
3. The system which is defined on the SYSTEM keyword (if specified for this console) on the CONSOLE statement in CONSOLxx:

   ```
   CONSOLE  DEVNUM(devnum)
            UNIT(unittype)
            NAME(cnname)
            SYSTEM(sysname)
   ```

4. The system on which the command executes

VARY CN Examples: Following are some examples of the use of the VARY CN command:

- The first example does a vary online of a console named TAPCON. The system chosen as the one on which TAPECTL is activated is determined by the steps shown in the section “VARY CN Command” on page 63.

  ```
  VARY CN(TAPCON),ONLINE
  ```

- The second example shows the command to vary online to system SC47 a console named CONS1.

  ```
  VARY CN(CONS1),ONLINE,SYSTEM=SC47
  ```

- The last example shows console CONS2 being varied offline, wherever it is attached in the sysplex. If a console device has not been active previously in the sysplex, and the SYSTEM keyword is not specified on the VARY command, then MVS uses the CONSOLE statement SYSTEM keyword to determine where to activate the console.

  ```
  VARY CN(CONS2),OFFLINE
  ```

VARY ONLINE/OFFLINE Command: VARY ONLINE and VARY OFFLINE should still be used to manipulate devices on a single system. MCS consoles can be brought online or offline with these commands, but they should be identified in the command by their device numbers. You should use the VARY CN(console_name) command for manipulating console-capable devices by their names. You are encouraged not to use the prefix “O-” to identify printer devices in any VARY ONLINE/OFFLINE commands.

FORCE Keyword: MVS/ESA SP 5.1.0 introduces the FORCE keyword on the VARY *device* ONLINE command. It is not valid for use with VARY OFFLINE. This keyword allows the operator to explicitly override the device state and bring

online to a single system a device which is being kept offline by a configuration manager.

MVS/ESA SP 5.1.0 also introduces the device state: “device being kept offline by a configuration manager.” This device state allows an installation to keep a shared device offline to all systems more easily. This may be required, for example, when essential maintenance is being performed on the device.

The following command makes devices 282 and 283 available for a system, even if they are being kept offline by a configuration manager:

```
VARY (282,283),ONLINE,FORCE
```

The FORCE keyword brings the specified device back online, even if the device is being kept offline by a configuration manager. It allows an installation to coordinate single-system vary device operations with vary device operations performed by a configuration manager, such as the ESCON Manager. If an operator attempts to vary online a device that is in this state, MVS rejects the command.

DISPLAY CONSOLE Command: The output from the DISPLAY CONSOLES command now includes information about the default system upon which an inactive console is activated. The default system is either:

- The last system on which the console was active
- The SYSTEM value specified in the CONSOLxx parmlib member

The status of the console, including whether it is active or inactive, is shown within the COND= field in the output from the DISPLAY CONSOLES command. See *MVS/ESA System Commands* for more details on the DISPLAY CONSOLES and VARY operator commands.

3.7 Operations Services

The enhancements to operations services in MVS/ESA SP 5.1.0 are as follows:

- A SYSOPS component trace function
- A HARDCOPY option on the OPERPARM parameter of the MCSOPER macro
- WTOR enhancements

3.7.1 SYSOPS Component Trace

MVS/ESA SP 5.1.0 introduces the component trace called SYSOPS, which allows you to collect diagnostic information about the operations services component trace (OPS). The OPS component, synonymous with the console services component of MVS, is used for tracing events that affect the console address space.

The SYSOPS component trace provides for a way for MVS components to have increased event-related data capture and diagnostic capability in a single system or a sysplex environment. Each component that uses the component trace service has set up its trace in a way that provides the unique data needed for that component. The following specifications are required to support this component trace function:

- CTIOPS00 parmlib member

- CTRACE parameter in the CONSOLxx parmlib member
- CT option on TRACE operator command
- Dynamic sizing of trace buffers

To provide support for the sysplex environment, the following component traces have been updated:

- SYSJES
- SYSOMVS
- SYSWLM
- SYSXES

IBM supplies a default CTIOPS00 member for the SYSOPS component trace, but does not supply a sample CONSOL00 parmlib member. Specify CTIOPS00 as the default on the CTRACE parameter of the INIT statement of CONSOLxx member. See *MVS/ESA SP V5 Initialization and Tuning Reference* for further details.

The CTIOPS00 parmlib member contains the default parameters for the OPS trace. The default options in the parmlib member are:

```
TRACEOPTS
      ON
      BUFSIZE(64K)
```

The CTIOPSxx member is pointed to by the entry in CONSOLxx parmlib member:

```
CTRACE(CTIOPS00)
```

The defaults for the OPS trace allow it to only collect information about unusual events (exceptions). This ensures that you will have trace data if a problem occurs, but with minimal overhead from the tracing facility. The system collects trace entries in a 64KB buffer. You can change the size of the trace buffers dynamically using the operator command:

```
TRACE CT,BUFSIZE=nnnnK  or nnnnM
```

You can also collect trace entries in a trace data set rather than to trace buffers.

3.7.1.1 Defining Trace Entries

If you wish to define different trace entries, then you need to add them to an appropriately named parmlib member. The naming convention for the parmlib member is CTncccxx, where:

n The source of the member:

- **I** IBM-supplied
- **U** User-supplied

ccc Component being traced

xx Any two characters for the member

For example, CTIOPS00 is the IBM-supplied default member for tracing the operations services (OPS) component events, and the member number is 00.

Defining a Parmlib Member: The following example illustrates how to specify the external writer. It assumes that there is a external writer procedure called OPSWTR in SYS1.PROCLIB and member CTUOPS01 in SYS1.PARMLIB, which contains the parameters:

```
TRACEOPTS
    WTRSTART(OPSWTR)
    ON
    WTR(OPSWTR)
    OPTIONS('MESSAGES')
    ASID(1)
    JOBNAME(PAYROLL)
    BUFSIZE(128K)
```

The external writer option prevents the trace entries from being overwritten. The trace buffer size specified at IPL can be adjusted while the system is running. You should use existing procedures in SYS1.PROCLIB or add your own procedure to call the external writer. To use the external writer, enter the command:

```
TRACE CT,,COMP=SYSOPS,PARM=CTUOPS01
```

Component tracing is more fully described in *MVS/ESA SP V5 Diagnosis: Tools and Service Aids*.

Formatting OPS Component Traces: You can look at the OPS trace entries by formatting them with IPCS. You use the IPCS subcommand CTRACE to process entries located in either:

- A buffer in an SVC or stand-alone dump, or
- A trace data set

Specify the appropriate data source, and then issue the command:

```
CTRACE  COMP(SYSOPS)  OPTIONS((MESSAGES))
```

In the output from this subcommand, IPCS labels each trace entry with one of the options WTO or MSGDLVRY.

3.7.1.2 TRACE Operator Command

An example of the TRACE operator command that uses the CT option follows:

```
TRACE CT,ON,PARM=mem,COMP+SYSOPS
```

Where:

PARM=mem This points to the relevant parmlib member that contains the trace parameters. If *mem* is not specified, the operator is prompted to enter the trace parameters.

COMP=SYSOPS This specifies that the component to be traced is the operations services component.

You can specify additional trace options for the OPS component either in the parmlib member, or by issuing the TRACE CT operator command and entering parameters in response to the WTOR. The options are the same in both cases, and are defined as follows:

Four additional options can be specified either in the CTIOPSxx parmlib member or on the reply to the TRACE CT command (message ITT006A):

WTO Traces the effects of MPFLSTxx, the user exits, and the SSI on messages content and attributes.

MSGDLVRY Traces events for WQE processing, MCS console message queueing and extended MCS console message processing.

MESSAGES This option includes the WTO and MSGDLVRY options.

SYSPLEX Traces events for XCF signalling, sysplex serialization services, sysplex clean-up processing and the manipulation of various queues.

An example that specifies a parmlib member:

```
TRACE CT,ON,COMP=SYSOPS,PARM=CTAOPS01
```

An example with no parmlib member:

```
TRACE CT,ON,COMP=SYSOPS
```

An example of the reply ID message and the response using one of the additional options mentioned above:

```
*nn,ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND

R nn,OPTIONS=(MESSAGES)
```

See *MVS/ESA SP V5 System Commands* for further details about the TRACE CT command and *MVS/ESA SP V5 Diagnosis: Tools and Service Aids* for further details about the parameters used for each component trace.

3.7.2 HARDCOPY and Extended MCS Consoles

Changes in MVS/ESA SP 5.1.0 to support extended MCS consoles and the HARDCOPY log are:

- The definition of extended MCS consoles and use the MCSOPER macro
- A HARDCOPY parameter on the OPERPARM parameter list of the MCSOPER macro to allow MCSOPER users to receive the hardcopy message set

An extended MCS console is a program that acts as a console. It can issue MVS commands, and receive command responses and unsolicited message traffic. There are two ways to use extended consoles:

- Interactively through IBM products such as TSO/E and NetView
- Through a user-written application program. Examples of application program uses are:
  - Receiving automated message traffic
  - Defining a unique presentation service for messages to consoles
  - Defining more than 99 consoles to a system or sysplex

3.7.2.1 MCSOPER Macro and Extended MCS Consoles

To establish a program as an extended MCS console, the program must issue the MCSOPER macro.

```
MCSOPER   REQUEST=ACTIVATE
          NAME=CONS1              CONSOLE NAME
          CONSID=(2)              CONSOLE ID
          TERMNAME=(4)            TERMINAL NAME
          MCSCSA=(3)              CONSOLE STATUS AREA
          MCSCSAA=(3)             ALET FOR CONSOLE STATUS AREA
          MSGECB=MESSAGE_ECB      MESSAGE ECB
```

Once activated as an extended MCS console, a program can receive messages and command responses by issuing the MCSOPMSG macro and can issue

commands by issuing the MGCRE macro. Unlike a standard MCS console, the extended MCS console can control which command responses it receives.

To issue commands, a program must issue the MGCRE macro.

```
MGCRE     TEXT=TEXTAREA          TEXTAREA THAT CONTAINS THE COMMAND
          CONSID=(2)             CONSOLE ID
          CART=USER_DEF_CART     COMMAND/RESPONSE TOKEN
          MF=(E,LISTADDR)        ADDRESS OF PARAMETER LIST
```

To receive messages and command responses, a program must issue the MCSOPMSG macro. Parameters on the MCSOPMSG macro enable a program to receive specific types of messages.

```
MCSOPMSG  REQUEST=GETMSG
          CONSID=(2)             CONSOLE ID
```

In a sysplex, an extended console application can receive messages from any system or systems in the sysplex, or can send commands to any system or systems in the sysplex. There is no system-imposed limit on the number of extended MCS consoles.

3.7.2.2 OPERPARM HARDCOPY Parameter

The HARDCOPY option on the OPERPARM parameter of the MCSOPER macro allows users of this macro to receive the MVS hardcopy message set. This facility can be used by applications to augment or replace the system log. By allowing the collection of all the messages in the MVS hardcopy message set from one or more systems in a sysplex, the MCSOPER macro can be used to create customized sysplex-wide system log applications.

There are two set of hardcopy messages that are affected by parameters of the CONSOLxx parmlib member or the syntax of the VARY HARDCOPY command. The two sets are:

Hardcopy message set The messages to be recorded on the log. The hardcopy message set is the set of messages as defined by the CONSOLxx member or modified by the VARY HARDCOPY command. You can specify the routing code and the kind of messages (commands, responses or status display) to be included in the hardcopy message set.

Log The installation-managed hardcopy medium where the hardcopy message set is recorded. The hardcopy medium is an MCS printer or SYSLOG that receives the hardcopy message set.

Messages in the hardcopy message set are either command responses or other types such as broadcast messages, messages generated by compilers, and so on. The hardcopy message set is the set of messages that was formerly received by SYSLOG only.

These are only terminology changes, without affecting or changing the definition of the CONSOLxx member or the VARY HARDCOPY command.

Note: The device number can now be specified with four digits, preceded by a slash if necessary to avoid confusing the device number with a device type. For more information, see 4.7, “4-Digit Device Support” on page 105.

3.7.2.3 JES3 Considerations

If an MCSOPER macro user specifies the HARDCOPY parameter on the OPERPARM parameter of the MCSOPER macro in a JES3 environment, the user may not receive the full hardcopy message set. JES3 manages its own "hardcopy log" and marks messages for no hardcopy to bypass any non-JES3 hardcopy logs. Marking a message for no hardcopy excludes the message from this HARDCOPY attribute processing. However, during the period when the "hardcopy log" is inactive, JES3 does not mark messages as no hardcopy. Therefore, any MCSOPER users who have specified the HARDCOPY parameter during the same period receive the hardcopy message set. Moreover, if the MCSOPER macro user specified any routing codes or other message attributes, it may be possible to receive some of the messages belonging to the hardcopy message set in a JES3 environment when those messages have attributes that match the ones specified by the MCSOPER macro user.

For more information on the MCSOPER macro, see *MVS/ESA SP V5 Auth Assembler Services Reference ALE-DYN*. Sample code that shows how to use an extended MCS console resides in SYS1.SAMPLIB member EXTMCS.

3.7.3 WTOR Enhancements

The changes to WTOR processing only apply to a sysplex in which all systems are at a minimum level of MVS/ESA SP Version 5. In a mixed version sysplex, WTOR reply IDs are the same as before MVS/ESA SP Version 5. The following WTOR enhancements have been made in MVS/ESA SP Version 5:

- WTORs are assigned reply IDs.
- Minimum value for RMAX in a sysplex is 99.
- Action message sequence numbers changed.

3.7.3.1 WTOR Reply IDs

Every Write to Operator with Reply (WTOR) has a unique number or identifier (ID). This ID is used to link the WTOR with a reply from the operator. Reply IDs are unique within a single MVS system or across a sysplex of multiple systems. In a sysplex, the operator can reply to a WTOR from anywhere within the sysplex. The maximum reply ID value is specified in the CONSOLxx member of SYS1.PARMLIB on the RMAX parameter of the DEFAULT statement.

3.7.3.2 RMAX Enhancements

Prior to MVS/ESA SP 5.1.0, RMAX defaulted to 99, but you could specify a value up to 9999. The value of RMAX could not be changed unless CONSOLxx was changed, and an IPL was required to implement the change.

With MVS/ESA SP Version 5, the minimum value of RMAX is 99 (the same maximum value of 9999 still applies), and the current setting of RMAX can be changed by operator command as follows:

```
K M,RMAX=5000
```

The command increases RMAX to 5000. The value of RMAX can be displayed using the operator command:

```
K M,REF
```

3.7.3.3 Action Message Changes

Action message sequence numbers are changed in MVS/ESA SP 5.1.0. They are now in the range:

```
1sss to 4294967sss
```

where sss is a unique decimal number generated by each MVS system.

3.8 Sysplex Device Drain

Sysplex device drain is a change in the vary offline processing design. In systems prior to MVS/ESA SP 5.1.0, when a VARY OFFLINE command is issued for a device, the device is first placed in pending offline state, and goes offline only when all the jobs that allocated the device have deallocated the device. But when a device is in a pending offline state, it is still eligible for certain allocation requests; so there can be a considerable delay between the time a VARY OFFLINE command is issued, and the time that the device goes offline.

In a sysplex environment, where all the data is shared among the systems, and where ESCON Manager V1 R3 is used as a single point of control for sysplex-wide operations, it is required that the time needed to vary a device offline be as short as possible. Also in a nonparallel sysplex environment, it is desirable that a device is not eligible for allocation once a VARY OFFLINE command is issued. The point here is to favor a system management issue (the vary offline operation) over a job issue (the job requiring the pending offline device).

In MVS/ESA SP 5.1.0, the vary offline process is favored over the requests to allocate the device. However, the operator and an installation exit are still given the chance to override the system′s decision, and let the allocation request be satisfied by allocating the pending offline device.

3.8.1 VARY OFFLINE Processing

For releases prior to MVS/ESA SP 5.1.0, when a device is in pending offline status, any specific volume request or any demand request for that device is satisfied by allocating the pending offline device to the requestor. These two types of allocation requests are very common. A specific volume request is one in which the volume serial information is provided in one of the following ways:

- In the JCL DD VOL=SER= keyword
- In the equivalent Dynamic Allocation Text Unit
- In a catalog entry for the data set being allocated

A demand request is one in which a particular unit is requested by specifying either:

- The JCL DD UNIT= keyword
- The equivalent dynamic allocation text unit

With MVS/ESA SP 5.1.0, once a device is in pending offline state, it is considered as if it is already offline for any kind of request.

For any allocation request for a pending offline device, the system calls the allocated/offline device exit, and issues message IEF877E (IEF877E replaces message IEF290E).

3.8.2 VARY OFFLINE Example

The example message and reply shown in Figure 25 is in response to the situation where a volume MYDATA is mounted on a DASD device B40, and allocated to the job $USEVOL. A VARY B40,OFFLINE is issued by the operator. Unit B40 goes into a pending offline state, as it is still allocated to the job $USEVOL.

TSO user $991100 wants to browse a data set on the volume MYDATA. This results in a specific volume request for the volume mounted on B40.

Message IEF877E is issued as follows:

```
IEF877E $991100 NEEDS 1 UNIT FOR TSOPROC ISP15153
FOR VOLUME: MYDATA
         OFFLINE
         D=*0B40
------------------------------------------------------
*12 IEF238D $991100 - REPLY DEVICE NAME OR 'CANCEL'.
------------------------------------------------------
R 12,B40
```

Figure 25. VARY OFFLINE Processing Example

After the reply by the operator, the volume MYDATA, mounted on unit B40, is allocated to user $991100. With releases earlier than MVS/ESA SP 5.1.0, the device B40 would have been allocated to the user without any user exit or operator involvement.

The allocated/offline device exit is called by allocation whenever a request for an offline or pending offline device is made. This exit can decide about the actions to take for the job: make it wait, cancel it, allocate the volume, or let the operator decide. It is easy for the exit to distinguish a pending offline device from an offline device. The parameter list that the exit is passed includes the field, UXOVLSER, that contains a volser only if the involved device is in pending offline state. For offline devices this field is always zeroes.

When message IEF877E is issued for an allocation request for a pending offline unit, it shows a one-entry list, consisting of only the involved unit. The operator can communicate to the system the appropriate action to be taken by replying to message IEF238D.

For more information about the Allocated/Offline exit, refer to the *Conversion Notebook* and the *Installation Exits* manuals. For a complete explanation of message IEF877E, refer to the *System Messages* manual.

3.9 GRS Enhancements

MVS/ESA SP 5.1.0 introduces one small change to global resource serialization, an additional option for the RESMIL parameter. GRS supports a global resource serialization complex with more than eight systems by adjusting the residence time of the ring system authority message in a range that is based on the number of systems in the complex.

In MVS/ESA SP 5.1.0, RESMIL has an additional optional value, OFF. RESMIL(OFF) indicates that the RSA-message residence time is zero and that GRS is not to tune this value.

Part of the parmlib member GRSCNFxx, RESMIL has an IBM supplied default value of (10). Should you wish to change this value, the correct syntax of the RESMIL parameter is:

```
GRSDEF RESMIL(OFF) TOLINT(180) ACCELSYS(99)
```

More information on using the RESMIL parameter can be found in *MVS/ESA SP V5 Planning Global Resource Serialization*

3.10 Automation of IOS Messages

During IPL, the system tries to perform initialization of all devices. If a device is not available, the IPL is stopped and operator intervention is required. If many systems with shared devices are IPLing at the same time, it is likely that a DASD device would be temporarily inaccessible from a given system if it is reserved by another system. This could lengthen the IPL time dramatically when a large number of devices are shared among multiple systems.

This process is repeated for each device configured to the system, in device number order. When an I/O request is issued, a time-out interval of two seconds is set. If the device has not responded within two seconds, the system issues message IOS120A, or IEA120A, asking the operator to indicate if the IPL should continue without the device (R 0,CONT), or if the system should wait for the device to be available (R 0,WAIT). When one of these messages is issued, the validation process is stopped.

The two second time-out interval, and the serial validation of the devices make this a time consuming process, with the potential for requiring a great deal of operator intervention.

MVS/ESA SP 5.1.0 enhances the I/O validation process through the function called IOS automate. During NIP and master scheduler initialization, when the IOS MIH has not yet initialized, IOS automate helps to reduce the need for operator intervention. The major design changes are:

- IOS now validates a group of devices concurrently; the I/O validation is now performed in parallel.
- A time-out interval of 15 seconds is set for the completion of the group of devices being validated. If after 15 seconds any of the devices in the group have not completed the validation process, IOS issues the message IOS123I, listing the devices in the group that have not been initialized. The time-out interval is consistent with the default MIH interval for DASD devices (which is 15 seconds).

Message IOS123I may be followed by either message IOS120A or IEA120A. If the operator replies WAIT in response to IOS120A or IEA120A, then IOS waits for an additional 15 seconds. If a device has still not responded, the operator is reprompted with message IOS124A, asking the operator whether to wait for additional 15 seconds, or to continue with the device.

This design eliminates the IOS120A and IEA120A messages that were unnecessarily issued for devices temporarily unavailable due to contention.

Additionally, by performing the I/O validation in parallel on a group of devices, the validation of other devices is not delayed by contention or errors on a single device.

For more information on the messages, refer to *MVS/ESA SP V5 System Messages* and *MVS/ESA SP V5 Conversion Notebook*.

The following is an example of the sequence of messages and responses that might occur during IPL.

```
IOS123I WAITING FOR RESPONSES FROM THE FOLLOWING DEVICES:
0201,0120,40A,31F

*03 IOS120A DEVICE 0201 SHARED.REPLY "CONT" OR "WAIT"
R 03,WAIT

*04 IOS124A STILL WAITING FOR RESPONSE FROM DEVICE 0201.
TOTAL WAIT TIME IS    30 SECONDS.REPLY "CONT" OR "WAIT"
R 04,WAIT

*05 IOS124A STILL WAITING FOR RESPONSE FROM DEVICE 0201.
TOTAL WAIT TIME IS    45 SECONDS.REPLY "CONT" OR "WAIT"
R 05,CONT
```

3.11 SMF Enhancements

MVS/ESA SP 5.1.0 allows a systems programmer to define multiple system identifiers in a single SMFPRMxx parmlib member. Previously, the restriction was to identify a unique system identifier per system. An installation can use a single SMFPRMxx parmlib member to define all the systems that are in a sysplex.

In the SMFPRMxx parmlib member, the SID parameter can be coded to specify:

- The system identifier to be used in SMF records
- A system identifier that uses the processor serial numbers specified at IPL
- A system identifier that uses the SYSNAME parameter defined in the IEASYSxx parmlib member
- A system identifier that uses a part of the SYSNAME parameter defined in the IEASYSxx parmlib

Another restriction that has been removed is the naming conventions of the SMF data sets. Previously, installations were restricted to SYS1.MANxx data sets. Although the number of data sets that were allowed to be defined in the past had been increased, the naming conventions were restrictive, which could have led to confusion in a multisystems environment. In MVS/ESA SP 5.1.0, this removes the requirement of SMF data sets having to have a prefix of SYS1.MAN. It expands the maximum length of the data set name to 44 characters and can be any format. However, it is recommended that the SMF data set name has an identifier associated with the system that is recording the data on the data set.

SMF data sets can be defined with symbolic names. This simplifies the naming process and allows an installation to create unique SMF data sets for the systems in the installation. Symbolic parameters may be used to substitute the

SID parameter, defined in SMFPRMxx. Another alternative is to use symbolic parameters in the SYSNAME parameter, defined in IEASYSxx.

These options simplify the administration tasks in a multisystem environment. An example of different options are listed in Figure 26.

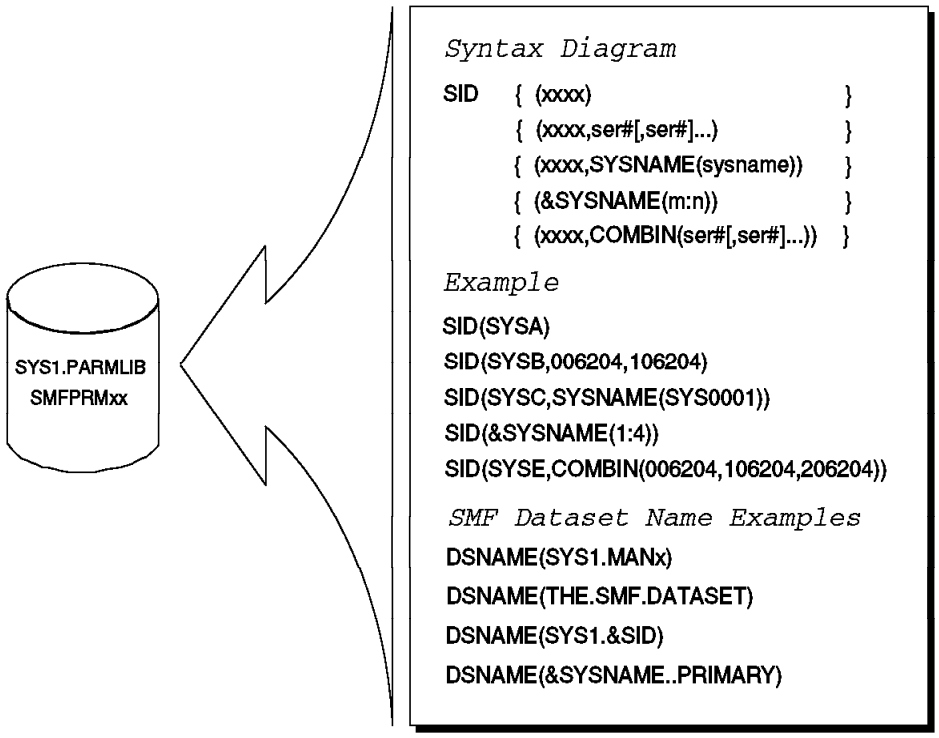


Figure 26. An Example of Different SMF Data Set Naming Conventions

For more details on the options for use in the SMFPRMxx parmlib member, please refer to *MVS/ESA SP V5 Initialization and Tuning Reference*.

Chapter 4. System Management Improvements

Many installations need to ensure that their system and its services are available and operating to meet service level agreements. Installations with 24-hour, 7-day operations need to plan for minimal disruption of their system activities. With the advent of sysplex many installations have replicated environments with similar setups on each of the images. This chapter discusses the many functions in MVS/ESA SP Version 5 that address these complexities and make easier the task of systems management. These functions include:

- Symbolic substitutions
- Dynamic subsystem interface
- Shared master catalog
- Job support for started tasks
- I/O Path validation
- Hardware configuration definition enhancements
- 4-digit device support
- Display RESERVE status
- Dynamic exits

4.1 Symbolic Substitution Overview

A sysplex environment is generally a replicated environment with a similar setup on each of the images. However, despite this similarity, there are many things that may be unique for each image. For example, each image in a sysplex must have a unique system name. Job names for multisystem applications should be unique to simplify problem determination; if unique names are not used, this could result in confusion (for example where did the duplicate named work originate). Data sets may need to be unique for each system (for example PAGE data sets, CTrace); however, jobs that access these data sets may need to run on each system, resulting in unique JCL for each image.

The multiple sets of JCL, multiple parmlib members and the processes to handle these tasks, result in increased maintenance costs and the possibility of errors occurring when changes are made or a new system is brought into the complex. Symbolic substitution is used in SYS1.PARMLIB member parameters, JES2 initialization parameters, MVS commands, and started task JCL. The intent of this support is to enable the use of a single parmlib member, a single procedure or a single command to handle multiple instances across the sysplex.

4.1.1 System Symbols

System symbols act like variables in a program; they can take on different values, based on the input to the program. When you specify a system symbol in a parmlib member the system symbol acts as a “place holder.” Each system that shares the definition replaces the system symbol with a unique value during initilization.

There are two distinct sets of symbols; they are discribed in the following sections.

4.1.1.1 Static System Symbols

Static symbols are symbols that do not vary the contents across an IPL.

Static system symbols have two types.

- System-defined static system symbols

  These symbols have their names defined to the system. Your installation defines substitution texts or accepts system default texts for static system symbols. These static system symbols are the following:

  ```
  &SYSCLONE
  &SYSNAME
  &SYSPLEX
  ```

- Installation-defined static system symbols

  These static system symbols are defined by you. You specify their names and substitution texts in SYS1.PARMLIB. You are allowed to specify a maximum of one hundred installation defined static system symbols.

4.1.1.2 Dynamic System Symbols

Dynamic system symbols are symbols whose substitution text can change at any point in an IPL. Dynamic system symbols represent values that can change often, such as dates and times. A set of dynamic system symbols are defined to the system. No additional dynamic system symbols can be added by you.

Please see *MVS/ESA Initialization and Tuning Reference* for further information regarding the definition of System symbols as well as a list of dynamic and static system symbols.

4.1.2 Parmlib Support

Support is provided for use of a set of system symbolics in many installation parameters. When these symbolics are used, a single installation definition (for example, a parmlib member) could be shared among multiple systems. The symbolics will resolve to unique specifications on the system using the parameters.

JES2 initialization parameters also support the use of the system symbolics. Please refer to *MVS/ESA SP-JES2 Version 5 Implementation Guide* for the JES2 support.

4.1.3 MVS Commands and Procedures

System symbolics are also useful in the operations area. A particular application may run on several images. One of the goals of replication is to use a common set of source JCL to initialize the application but retain the uniqueness where required. The most efficient way to do this is to have a single set of source JCL that uses symbolics for those parameters that must be unique per image and initiate the application as a started task or started job.

MVS/ESA SP 5.1.0 provided base support that allowed a unique jobname to be associated with a single set of source JCL by adding a JOBNAME= parameter to the start command. Support was also provided that allowed the source JCL for a started task to be a job. Although only a subset of the JOB card parameters were allowed, it provided the level of accounting, security and output control that was required by many customers in order to use started tasks.

In MVS/ESA SP 5.2.0, support is now provided by allowing the use of system symbolics in:

- The source JCL for started tasks and TSO logon procedures
- MVS commands
- Dynamic allocations

With this support a single set of source JCL can be started on all systems in a sysplex with a single command (either via a common COMMNDxx member or via a ROUTE *ALL command) and resolved to different jobnames on each image.

For example, in a sysplex with systems S1, S2 and S3: If a common COMMNDxx member contains the following command:

```
S CICS,JOBNAME=CICST&SYSNAME,HLQ=&SYSNAME,.....
```

The jobname will be:

```
CICSTS1 on S1
CICSTS2 on S2
CICSTS3 on S3
```

The capability of using symbolics directly in the source JCL for started tasks is particularly useful for instances, such as CTRACE, where unique datasets are needed but no parameters are supplied for this purpose.

```
//IEFPROC   EXEC  PGM=ITTTRCWR
//SYSPRINT  DD    SYSOUT=A
//TRCOUT01  DD    DSN=SYS1.JESXCF1.&SYSNAME,DISP=SHR
//TRCOUT02  DD    DSN=SYS1.JESXCF2,&SYSNAME,DISP=SHR
```

This procedure for the started task IXZCTW may now be used on different systems and each system will resolve the “&SYSNAME” symbolic.

The MODIFY command is also enhanced to allow a trailing wildcard in the jobname. This allows a set of commonly named instances on one or more systems in the sysplex to be modified with a single command when used in conjunction with the ROUTE command. For example:

```
ROUTE *ALL,MODIFY CICST*,....
```

4.1.4 Related Support

The MVS/ESA SP 5.1.0 support that allows the source JCL for started tasks to contain a JOB statement and JOB level controls statements requires the source JCL to reside in a procedure library in the master scheduler JCL concatenation. Changing MSTJCLxx entailed an assembly and link-edit to add a procedure library to the concatenation. With MVS/ESA SP 5.2.0, MSTJCLxx may be in SYS1.PARMLIB, making a MSTJCLxx user modification unnecessary. See *MVS/ESA Initialization and Tuning Reference* for further information regarding the details.

4.1.5 Implementation of Symbolics

Before MVS/ESA SP 5.2.0, multiple LOADxx members had to exist in SYSx.IPLPARM or SYS1.PARMLIB to IPL multiple images that pointed to different IEASYSxx members via the SYSPARM statement in order to use a different SYSNAME in IEASYSxx. With MVS/ESA SP 5.2.0, it is now possible to specify one LOADxx member pointing to a new parmlib member (IEASYMxx). IEASYMxx

specifies the symbols in use as well as the different IEASYSxx members to use for each image as shown in Figure 27.

In Figure 27 three systems are shown pointing to one common LOADxx member in SYSx.IPLPARM or SYS1.PARMLIB. This LOADxx member then points to one IEASYMxx member in SYS1.PARMLIB. In the IEASYMxx member, four SYSDEF statements are shown. The first SYSDEF statement is a global SYSDEF statement, as no HWNAME, LPARNAME or VMUSERID is specified. The next three SYSDEF statements apply to the three systems shown. As there is a SYSPARM statement in the global SYSDEF stating the default IEASYSxx member to use (IEASYS00), this is the IEASYSxx member that system SC52 and SC53 will use. The local SYSDEF statement for SC50 specifies a SYSPARM statement, and therefore, system SC50 will use IEASYS50 and IEASYS00.

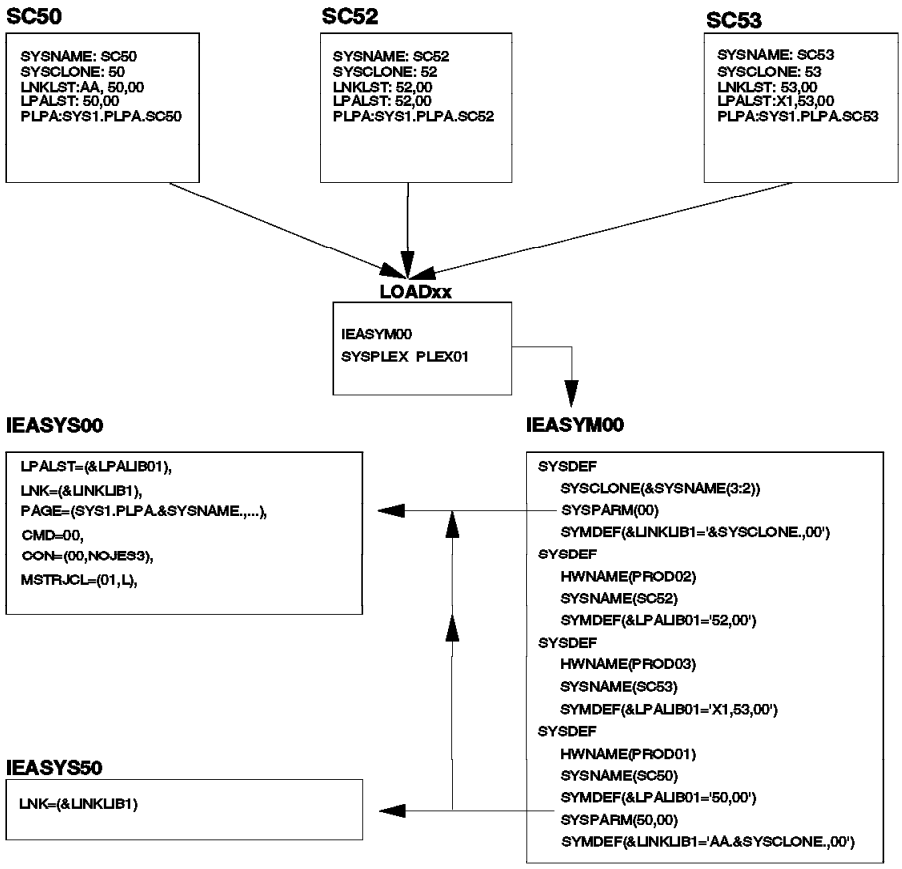


Figure 27. View of Symbolic Support

A sample program (IEASYMCK) is provided in SYS1.SAMPLIB to read members from a data set and substitute any system symbols found. The system symbols used for the substitution are the system symbols in effect on the system on which the job is run. The shortcoming in this test is that the symbols have to be predefined and then the system must be IPLed prior to running the program.

To share SYS1.PARMLIB or SYSx.IPLPARM members do the following:

- Identify resources that are more easily managed by a naming convention that supports symbols.

- Determine appropriate naming conventions.
- Update SYS1.PARMLIB and SYSx.IPLPARM to implement naming conventions using symbols. See *MVS/ESA Initialization and Tuning Reference* for more information.
- Identify, communicate and implement operational changes resulting from the new naming convention and the new symbolic support.

The different resources for naming conventions could consist of the following:

- The system name(&sysname)
- The SMF identification
- The specific logical partition name(LPAR)
- Specific job and started task (STC) names
- Specific job and STC JCL
- MSTJCLxx used across the parallel sysplex
- SYSx.IPLPARM and SYS1.PARMLIB members that can be shared
- The data set names used for SMF, LOGREC, VIO journaling, SWAP and Page

Hints and tips for naming conventions across the parallel sysplex:

- Try to keep the system name to four alphanumeric characters. If the system name is kept short, it requires less typing when commands are issued across the parallel sysplex.
- Try to keep the system name, SMF ID and JES2 member name the same.
- Try and have a unique two letter suffix on the system name that can be used for the &SYSCLONE symbolic variable, for example SYSA or SYSB. The &SYSCLONE symbolic variable will then be SA or SB. This suffix can then be used for the suffix for specific SYS1.PARMLIB members. If a value is not assigned to &SYSCLONE, then the default value will be the last two letters in &SYSNAME. For example, if the following is specified in IEASYMxx for system SC52 and SYSCLONE is not specified,

```
SYSDEF
      HWNAME(PROD02)
      SYSNAME(SC52)
```

  then &SYSCLONE will have the contents of “52.”
- If a two letter suffix is decided upon, this could be used for all the system software, for example VTAM, CICS and MVS.

Once naming conventions have been decided upon SYS1.PARMLIB and SYSx.IPLPARM can be updated to reflect these.

To enable the use of one LOADxx member, the following changes were made to the different SYSx.IPLPARM and SYS1.PARMLIB members.

Changes made to LOADXX:

- The only difference between the sysplex images must be the SYSPARM statements in the different LOADxx members. Compare the current LOADxx members and ensure that this is true. An example of the current LOADxx members are shown in Figure 28 and Figure 29.

```
IODF     55 SYS1     LO6RMVS1 01
NUCLEUS  1
NUCLST   XX
SYSCAT   TOTCAT113CCATALOG.TOTICFM.VTOTCAT
SYSPARM  52
```

Figure 28. LOAD52 from System SC52

```
IODF     55 SYS1     LO6RMVS1 01
NUCLEUS  1
NUCLST   XX
SYSCAT   TOTCAT113CCATALOG.TOTICFM.VTOTCAT
SYSPARM  53
```

Figure 29. LOAD53 from System SC53

- Create an entry in LOADxx to point to the IEASYM member.

  In MVS/ESA SP 5.2.0 a new parameter was added to LOADxx: SYSPLEX. This parameter specifies the name of the sysplex in which this system participates. The sysplex name is also the substitution text for the &SYSPLEX system symbol. It is recommended that SYSPLEX is specified in LOADxx instead of in COUPLExx, as the substitution text for &SYSPLEX can then be used early on in system initialization. See *MVS/ESA Initialization and Tuning Reference* for further information.
- Remove the SYSPARM statement from LOADxx.
- Add a SYSPLEX statement to LOADxx.
- The new LOADxx member is shown in Figure 30.

```
IEASYM   (XX,L)
IODF     55 SYS1     LO6RMVS1 01
NUCLEUS  1
NUCLST   XX
SYSCAT   TOTCAT113CCATALOG.TOTICFM.VTOTCAT
SYSPLEX  WTSCPLX1 X
```

Figure 30. LOADXX that is Common for SC52 and SC53

A new member (IEASYMxx) was created in SYS1.PARMLIB:

- Add to IEASYMxx the symbolic definitions decided upon previously.
- Add SYMDEF statements to IEASYMxx which cater for each parallel sysplex image.
- Add a sysname statement to the appropriate SYMDEF entry in IEASYMxx.
- Add a SYSPARM statement to the appropriate SYMDEF entry in IEASYMxx.

  The new IEASYMxx member is shown in Figure 31.

```
SYSDEF          SYSCLONE(&SYSNAME(3:2))
                SYMDEF(&CMDLIST1='&SYSCLONE.,00')
                SYMDEF(&LNKLJES3='AA')
                SYMDEF(&LNKLIST2='00')
                SYMDEF(&LPALIST2='00')
                SYMDEF(&LPALSTJ1='AA')
                SYMDEF(&MLPALST1='AA')
SYSDEF          HWNAME(P101)
                LPARNAME(A1)
                SYSPARM(52)
                SYSNAME(SC52)
SYSDEF          HWNAME(P101)
                LPARNAME(A2)
                SYSPARM(53)
                SYSNAME(SC53)
```

Figure 31. IEASYMxx Created to Cater for SC52 and SC53

Changes made to IEASYS52 and IEASYS53:

- Remove the sysname statement from the relevant IEASYSxx members.
- Change the load parameters on the appropriate hardware consoles.
- IPL all the relevent sysplex images.

All the symbols decided upon previously are now in the sysplex images. The relevant parmlib members can now be changed to contain the symbolic statements. See *MVS/ESA Initialization and Tuning Reference* for a complete list of parmlib members that support substitution. After changing these members, run the sample program provided in SYS1.SAMPLIB (IEASYMCK) to verify that the parmlib members are correct.

Using one IEASYSxx member:

- Compare the different IEASYSxx members to see whether they are similar. Sample IEASYSxx (IEASYS52 and IEASYS53) members are shown in Figure 32 and Figure 33.

```
CMD=(52,00),
LNK=(AA,00),
MLPA=(AA),
LPA=(AA,00)
```

Figure 32. IEASYS52 Used for System SC52

```
CMD=(53,00),
LNK=(AA,00),
MLPA=(AA),
LPA=(AA,00)
```

Figure 33. IEASYS53 Used for System SC53

- Combine the IEASYSxx members and use symbolics for the differences. The new IEASYSxx member is shown in Figure 34.
- IPL all the relevent sysplex images.

```
CMD=(&CMDLIST1.),
LNK=(&LNKLJES3.,&LNKLIST2.),
LPA=(&LPALSTJ1.,&LPALIST2.),
MLPA=(&MLPALST1.)
```

Figure 34. IEASYSxx Combined and Used for Systems SC52 and SC53

To combine all the other members in SYS1.PARMLIB, first see *MVS/ESA Initialization and Tuning Reference* to confirm that system symbols are supported in the particular member. If the system symbols are supported, create a member using the system symbols. Again run the sample program provided to see whether the substitution is correct.

Proposed naming conventions for static symbols:

- Keep the SYMBOL name to a length of eight.
- When naming a symbol make it meaningful.
- Keep the same names across the parallel sysplex images.
- When using the symbol name, always code it with a period; see Figure 35.

```
CMD=(&CMDLIST1.),
CON=(&CONSOLXX.),
LNK=(&LNKLJES3.,&LNKLIST2.),
LPA=(&LPALSTJ1.,&LPALIST2.),
MLPA=(&MLPALST1.),
PROG=(&PROGLST1.),
SCH=(&SCHLST01.),
SMF=(&SMFLST01.),
SSN=(&SSNLST01.),
COUPLE=(&COUPLEXX.)
PLPA=(PAGE.&SYSNAME..LOCAL1,.........
```

Figure 35. Example for Coding Symbols in IEASYSxx

Notes to remember when defining symbols:

- The symbol name may not be longer than eight characters.
- A symbol may not be assigned a null content as shown in Figure 36.
- The length of the substitution text cannot exceed the length of the symbol name as shown in Figure 37.

```
        SYMDEF(&LNKLIST='')
```

Figure 36. Illegal Assignment for a Symbol Name

```
        SYMDEF(&A='ABCDEF')
```

Figure 37. Invalid Assignment of Substitution Text

A *DISPLAY* command is provided to display the current defined symbols in the parallel sysplex image. The result of the DISPLAY SYMBOLS command is shown in Figure 38.

```
D SYMBOLS
IEA007I STATIC SYSTEM SYMBOL VALUES 754
        &SYSCLONE. = 52
        &SYSNAME.  = SC52
        &SYSPLEX.  = WTSCPLX1
        &CLNLST.   = AA
        &CMDLIST1. = XX
        &CONNOJES. = NOJES3
        &CONSOLXX. = 00
        &COUPLEXX. = AM
        &IEASYSP.  = XX
        &LNKLIST2. = 00
        &LNKLJES3. = AA
        &LOGREC1A. = LOG
        &LOGREC2A. = STR
        &LOGREC3A. = EAM
        &LPALIST2. = 00
        &LPALSTJ1. = AA
        &MLPALST1. = AA
        &PROGLST1. = 00
        &SCHLST01. = 00
        &SMFLST01. = 01
        &SSNLST01. = XX
```

Figure 38. Results of DISPLAY SYMBOLS Command

4.1.6 Potential Problems

Please be aware of the following:

- If you specify a duplicate SYSCLONE value, as shown in Figure 39, unpredictable results can occur.

```
    SYSDEF          HWNAME(P101)
                    LPARNAME(A1)
                    SYSPARM(XX)
                    SYSNAME(SC52)
                    SYSCLONE(52)
    SYSDEF          HWNAME(P101)
                    LPARNAME(A2)
                    SYSPARM(XX)
                    SYSNAME(SC53)
                    SYSCLONE(52)
```

Figure 39. Duplicate SYSCLONE Values

  This can be prevented by specifying an option in the LOADxx member on the SYSPLEX statement.

- Be aware of the fact that if the command delimeter used for a specific system is "&," then the commands using symbolics will not work as intended.
- If there is a syntax error in IEASYMxx, MVS issues message IEA011A. A prompt is issued for a new member or to continue with the IPL without IEASYMxx support.

- When an IPL occurs and IEASYMxx is used and no local SYMDEF entry is found for that particular parallel sysplex image, the global SYMDEF defaults will be used. These might not give the desired results.

4.2 Dynamic Subsystem Interface

In MVS/ESA SP 5.1.0 and previous MVS/ESA systems, the subsystem interface only allowed an installation to define subsystems at an IPL, and activate or deactivate a subsystem by manipulating the system control blocks. Dynamic SSI in MVS/ESA SP 5.2.0 provides a set of authorized system services to allow installations to modify the subsystems defined to a system without requiring an IPL.

4.2.1 Dynamic SSI

There are now three methods of invoking the dynamic SSI services.

- Parmlib processing during an IPL

  If it is known that a subsystem will be required, the subsystem can be defined in an IEFSSNxx member of SYS1.PARMLIB. To enable dynamic SSI, the new format of IEFSSNxx must be used. IEFSSNxx changes from using positional parameters to keyword parameters. If you use the old format, then dynamic SSI will not be available for use.

  Conversion of IEFSSNxx

  To convert the old positional parameter form of IEFSSNxx to the new keyword parameter form of IEFSSNxx, use the sample REXX routine supplied in SYS1.SAMPLIB(IEFSSNxx). The old form of IEFSSNxx is shown in Figure 40, and the new form of IEFSSNxx is shown in Figure 41.

```
SMS,IGDSSIIN,'ID=TB,PROMPT=NO'                SMS
JES2,,,PRIMARY,NOSTART      PRIMARY SUBSYSTEM NAME
IRLM                        IMS RESOURCE LOCK MANAGER
JRLM                        SECONDARY SUBSYSTEM NAME FOR IRLM
PSP                         SUBSYSTEM NAME FOR BATCHPIPES
```

Figure 40. Old Form of IEFSSNxx

```
SUBSYS SUBNAME(SMS)       /* SMS */
  INITRTN(IGDSSIIN)
  INITPARM('ID=TB,PROMPT=NO')
SUBSYS SUBNAME(JES2)      /* PRIMARY SUBSYSTEM NAME */
  PRIMARY(YES)  START(NO)
SUBSYS SUBNAME(IRLM)      /* IMS RESOURCE LOCK MANAGER */
SUBSYS SUBNAME(JRLM)      /* SECONDARY SUBSYSTEM NAME FOR IRLM */
SUBSYS SUBNAME(PSP)       /* SUBSYSTEM NAME FOR BATCHPIPES */
SUBSYS SUBNAME(RACF)      /* RACF SUBSYS */
  INITRTN(IRRSSI00)
  INITPARM('#')
```

Figure 41. New Form of IEFSSNxx

- Macro invocation during program execution

  Dynamic SSI introduces a set of executable macros by which the new services can be requested. Invoking programs must be authorized

(APF-authorized or in supervisor state) to use any of the macro options except the subsystem query function. Refer to *MVS/ESA Using the Subsystem Interface* for examples.

- System commands

  The dynamic SSI command SETSSI can be used to control a subsystem if that subsystem supports the dynamic SSI command.

  The DISPLAY SSI command can be used to display information about any subsystem, dynamic or not, or about all subsystems.

4.3 Shared Master Catalog

Before MVS/ESA SP 5.1.0, you could either share a master catalog or a SYSRES volume among multiple images, but you could not do both. This restriction was caused by the requirement that certain data sets have fixed names, and that the data sets could not be shared between systems (notably, SYS1.LOGREC and SYS1.STGINDEX).

To share an IPL volume (SYSRES), you must:

- Have separate master catalogs for each system
- Explicitly catalog all fixed-name data sets to other volumes

With MVS/ESA SP 5.1.0, changes are made to allow sharing of both the master catalog and the system residence volume. To share a master catalog, you must catalog fixed data set names using "******" for the VOLSER and "0000" for the device type (UNIT=). This is known as an indirect catalog pointer. When the master catalog is searched, the indirect pointers make the system search the IPL volume for the specified data sets. Thus, all such data sets with indirect pointers in their catalog entry *must* be on the IPL volume.

To share both the IPL volume (SYSRES) and the master catalog among multiple MVS/ESA images, as shown in Figure 42, the following definitions are possible:

- Symbolic variables can be used as part of the system data set name.
- New parameters in SYS1.PARMLIB(IEASYSxx).

In previous releases of MVS/ESA, some data sets had unique names that are required to be cataloged in the master catalog. As each system had to have, for example, its own SYS1.LOGREC and SYS1.STGINDEX data set, it was impossible to share the master catalog. MVS/ESA SP 5.1.0 makes it possible to share a single master catalog across multiple systems, including systems in a sysplex. New capabilities simplify the setup of the master catalog and the sharing of the SYSRES volume. Installations can use symbolic names for these required data sets, thereby making the data sets unique to the systems sharing the master catalog. These symbolic names are specified in IEASYSxx located in SYS1.PARMLIB.

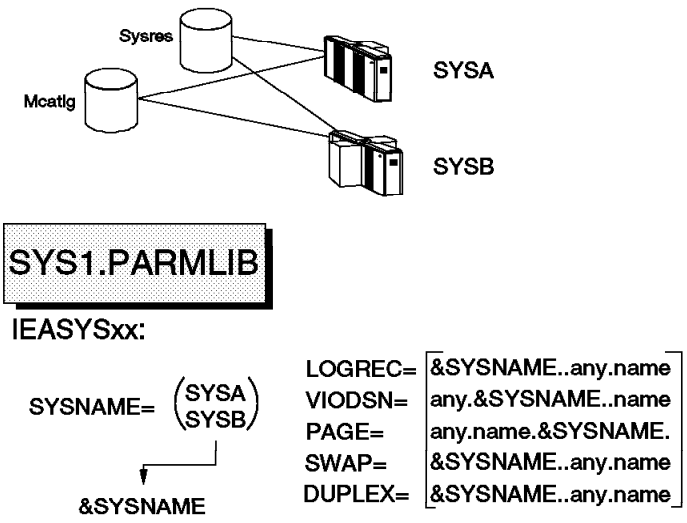


Figure 42. Shared Master Catalog and SYSRES

In the IEASYSxx parameters, you can specify unique values and data set names for the following data sets:

LOGREC Use the LOGREC parameter to specify the name of the logrec data set for error recording. SYS1.LOGREC is its default.

VIODSN Use the VIODSN parameter to specify the name of the VSAM data set that stores information about journaled VIO data sets. SYS1.STGINDEX is its default.

PAGE Use symbolic names in the page data set names to distinguish between the different systems that are sharing the master catalog.

SWAP Use symbolic names in the SWAP data set names to distinguish between the different systems that are sharing the master catalog.

DUPLEX Use symbolic names in the DUPLEX data set name to distinguish between the different systems that are sharing the master catalog.

NONVIO Use symbolic names in the NONVIO data set name to distinguish between the different systems that are sharing the master catalog.

Figure 43 illustrates the flexibility of the naming conventions in MVS/ESA SP Version 5 where two systems (SYSA and SYSB) share the master catalog. As can be noticed, the valid combinations are endless. It is advisable to maintain the SYS1 prefix as this would identify the data sets as system data sets.

Note: &SYSNAME is a symbolic variable introduced by MVS/ESA SP 5.1.0. It should not be confused with the SYSNAME parameter of IEASYSxx or IEASYMxx. The SYSNAME parameter was introduced in an earlier release of MVS/ESA to identify a system in a GRS or sysplex environment. Take care if you use the &SYSNAME symbolic as the HLQ, as this means that the system name is not available as an ALIAS. This is because the system data sets still have to be cataloged in the master catalog.

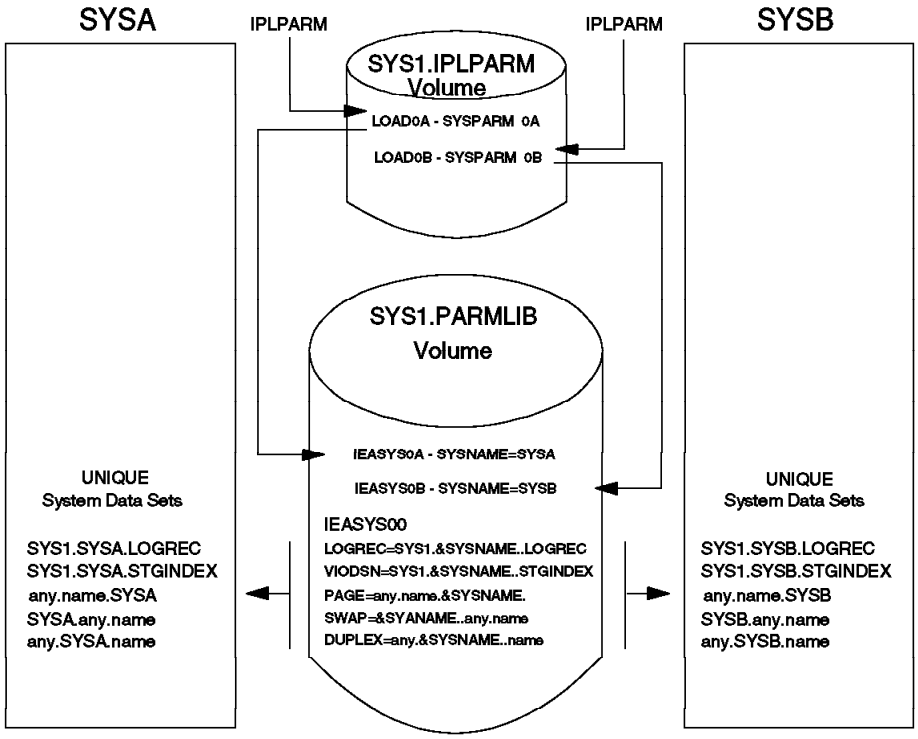


Figure 43. Use of the &SYSNAME Symbolic

As shown in Figure 43, during the IPL, IEASYS00 contains the common parameters. IEASYMxx contains the SYSNAME parameter introduced in MVS/ESA SP 5.2.0. The &SYSNAME is replaced with the name specified on the SYSNAME parameter of the IEASYMxx parmlib member. For more information, see the *MVS/ESA SP V5 Initialization and Tuning Reference* manual.

Two new keywords are added to define the LOGREC and VIO journaling data set:

- LOGREC=dsname
- VIODSN=dsname

Note: LOGREC=, VIODSN= and PAGE= are required parameters. If they are not specified, the operator receives messages that must be acted upon before the IPL process continues.

In any installation, whether single system, multiple unlinked processors, or sysplex, it makes practical sense to share as many system data sets as possible to reduce overall systems management effort.

Note: You do not have to share any data if you do not want or need to do so, but if you have a large or fast-growing installation with multiple images, it makes sense to try to share data. Certainly an n-way sysplex becomes a more manageable system structure if you do share your master catalog and SYSRES volume among the processors in your sysplex.

4.3.1 Examples of IEASYSxx Specifications

Each data set name must be a valid name consisting of a maximum of 44 characters, *after symbolic substitution is resolved*. Normal coding rules apply for characters that are valid in data set names.

- LOGREC=SYS1.LOGREC
- LOGREC=ANY.DATASET.NAME
- LOGREC=MYSYSTEM.LOGREC
- LOGREC=&SYSNAME..LOGREC
- LOGREC=LOGREC.&SYSNAME.

- VIODSN=IGNORE
- VIODSN=VIODSN.SYSTEMA
- VIODSN=SYS1.STGINDEX.&SYSNAME.
- VIODSN=&SYSNAME..ANY.DSNAME
- DUPLEX=SYSTEMA.DUPLEX
- DUPLEX=SYS1.DUPLEX
- DUPLEX=ANY.DSNAME.&SYSNAME.
- DUPLEX=&SYSNAME..DUPLEX

- NONVIO=(LOCAL1.SYSTEMA,LOCAL2.PAGE)
- NONVIO=SYS1.LOCPAGES
- NONVIO=ANY.LOCAL.&SYSNAME.
- NONVIO=(&SYSNAME..LOCALX,LOCALY.&SYSNAME.)

- PAGE=(PLPA.SYSTEMA,SYS1.COMMON,&SYSNAME..LOCALX)
- PAGE=(ANY.PLPA.&SYSNAME,ANY.COMM.&SYSNAME.,ANY.LOCL.&SYSNAME.)

- SWAP=(SWAP.DATA1,SWAP.DATA2)
- SWAP=SYS1.SWAP
- SWAP=(&SYSNAME..SWAPDS)
- SWAP=(ANY.SWAP.&SYSNAME.,MORE.SWAPD.&SYSNAME.)

4.3.2 Explicit References to LOGREC Specifications

Each program, procedure, parameter or JCL statement that explicitly references SYS1.LOGREC as the name of the LOGREC data set may need to be changed if you have used the facilities of MVS/ESA SP 5.1.0 to give LOGREC unique data set names on each of your MVS images.

- Remove the LOGREC data set name from the systems exclusion resource name list (if you use GRS in a sysplex or GRS ring).

Change anything that explicitly references SYS1.LOGREC by looking at the following functions:

- Procedures and programs
  - IFCDIP00
  - EREP (on SERLOGDD statement)
  - IFCOFFLD (on SERLOGDD statement)
- PARMLIB members
  - GRSRNLxx
- SAMPLIB members
  - ARCSTRST

  - ISGGRNLS
- Other considerations
  - SMP/E
  - CBIPO or similar

4.3.3 LOGREC and the MVS System Logger

In a sysplex environment the number of logs become extremely difficult to handle. A new facility is provided whereby the logs can be written to the MVS system logger. This capability enables LOGREC to write the EREP records to one specific data set using MVS system logger. Refer to *MVS/ESA Version 5 Sysplex Implementation Guide* for the implementation of LOGREC using MVS system logger.

4.4 Job Support for Started Tasks

MVS/ESA SP 5.1.0 allows you to assign job names to started tasks by using a job as the source for the started task rather than a procedure. This allows you to track multiple instances of the same started task (a task only operators could perform before). You can also associate job-level characteristics with the started task. For example, job support for started tasks has the following enhancements:

- Started tasks can have job cards:
  - Allows job-level JCL to be associated with a started task
- New started job library specifiable in the MSTJCLxx
- The MVS START command is enhanced:
  - A job name can be specified for existing started tasks
- Allows cloning of work elements

4.4.1 Job Cards in Started Tasks

If you choose to code a started task with a JOB statement, the rules are slightly different that the rules for other jobs:

- The member containing the job must have a job card in the first record.
- The statement must start with //.
- The jobname is one through eight non-blank characters.
- The jobname does not have to follow conventional JCL jobname rules (in terms of valid characters); however, if the jobname is not valid, it must be overridden by the JOBNAME parameter of the START command. If a name is not valid and is not overridden, a JCL error results.
- The jobname must be followed by at least one blank.
- JOB must follow the blanks after the jobname.
- JOB must be followed by at least one blank.
- The JOBNAME parameter of the START command can contain up to eight characters; it is therefore recommended that you ensure that the JOB statement contain enough room to accommodate eight-character names.

  The JOB card may have any valid name. The job name can be changed at START time by using the new JOBNAME parameter on the START command.

- Some JOB statement parameters are invalid for started jobs, such as:
  - Parameters that specify how a job is to be selected from the input queue have no meaning:
    - CLASS
    - RESTART
    - RD
    - TYPRUN

    In some cases, the invalid parameters cause the started job to be failed before execution. In other cases they are ignored.
  - Parameters that may create a potential security exposure are not allowed:
    - GROUP
    - PASSWORD
    - SECLABEL
    - USER

Note: If there is not enough space on the JOB card to place the job name specified on the START command, the START command fails with message IEE402I. Two new messages are issued when an invalid parameter on the JOB card has been coded. Message IEE401I is issued if the parameter is ignored. Message IEE399I is issued if the job is cancelled before execution. For an explanation of the messages, refer to *MVS/ESA SP V5 System Messages, Vol 4 (IEC-IFD)*

For more information on using a job as the source for a started task and the JCL considerations, please refer to *MVS/ESA SP V5 JCL Reference*.

The same criteria apply to JECL statements; those parameters related to input processing and those that could create a potential security exposure are rejected or ignored.

Note: JES3 does not support any JECL statements in a started task.

4.4.1.1 Job Card Considerations

You can specify accounting information on the JOB statement, which helps to reduce the number of SMF installation exits (or other installation-written programs) that are required for specifying accounting information.

You can control the started task output by specifying the MSGLEVEL on the JOB statement. For example, many installations purge all output from started tasks because of the volume of output. With the output control allowed within a job, you can specify to receive output only if something abnormal occurs with the started task. Also, all output from a started task can be directed to where it is required.

SYSIN data can be used to allow passing of parameters and data to associated programs in a started task.

4.4.2 Started Job Library

To create a job for a started task, you can use a data set that contains only jobs, or you can mix the jobs with procedures. If you want to keep the jobs and procedures separate, you can use the IEFJOBS data set and update your MSTJCLxx to include the IEFJOBS DD as follows:

```
//IEFJOBS  DD DSN=SYS1.STCJOBS,DISP=SHR
```

You can concatenate several data sets to the IEFJOBS ddname, exactly as you can do for the data sets concatenated to the IEFPDSI ddname.

If you do not want to update MSTJCLxx, you can still take advantage of this support by placing the jobs you want to start in one of the IEFPDSI data sets.

However, it is recommended to define the IEFJOBS ddname in MSTJCLxx for several reasons:

- By defining the ddname IEFJOBS, you can keep jobs and procedures separated. This allows jobs and the procedures they invoke to have the same name.
- Existing procedure libraries need not be modified.
- You need not worry about products that ship procedures that are placed in procedure libraries. If you modify a procedure to add the JOB card and a new version of the procedure is received, you do not lose your modifications.

With the introduction of the new IEFJOBS DD, it is important to note the following definitions:

IEFJOBS DD data sets must contain *only* jobs.

IEFPDSI DD data sets can contain both jobs and procedures.

JES JES procedure libraries must contain only jobs.

If the member to be started is found in IEFJOBS and it contains a procedure instead of a job, the START command fails with message IEE404I.

Again, it is advisable to use the IEFJOBS DD in the master JCL to keep those started tasks using jobs as their source separate from those using standard procedures. Procedures should still reside in a library pointed to by the IEFPDSI DD card in the MSTJCLxx member used at IPL time if they are to be started under the control of the master subsystem.

In the example below we see that the IEFJOBS DD card points to a data set called SYS1.STCJOBS. This name can be any installation chosen name.

```
MSTJCL05 CSECT
         DC    CL80'//MSTJCL05 JOB MSGLEVEL=(1,1),TIME=1440'
         DC    CL80'//         EXEC PGM=IEEMB860,DPRTY=(15,15)'
         DC    CL80'//STCINRDR DD SYSOUT=(A,INTRDR)'
         DC    CL80'//TSOINRDR DD SYSOUT=(A,INTRDR)'
         DC    CL80'//IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR'
         DC    CL80'//IEFJOBS  DD DSN=SYS1.STCJOBS,DISP=SHR'
         DC    CL80'//IEFPARM  DD DSN=SYS1.PARMLIB,DISP=SHR'
         DC    CL80'//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR'
         DC    CL80'//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR'
         DC    CL80'/*'
         END
```

The IEFJOBS DD statement defines the data set that contains job source JCL for started tasks. This data set can be a PDSE and, therefore, SMS managed. The data set may also be part of a concatenation.

Prior to MVS/ESA SP 5.2.0, MSTJCLxx had to be assembled and linked into SYS1.LINKLIB. With MVS/ESA SP 5.2.0 MSTJCLxx is allowed to reside in SYS1.PARMLIB. To create the new parmlib member MSTJCLxx, edit the current source for MSTJCLxx from samplib, and place that in SYS1.PARMLIB. The following example shows the new MSTJCLxx member in SYS1.PARMLIB.

```
//MSTRJCL  JOB MSGLEVEL=(1,1),TIME=1440
//         EXEC PGM=IEEMB860,DPRTY=(15,15)
//STCINRDR DD SYSOUT=(A,INTRDR)
//TSOINRDR DD SYSOUT=(A,INTRDR)
//IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR
//IEFJOBS  DD DSN=SYS1.STCJOBS,DISP=SHR
//IEFPARM  DD DSN=SYS1.PARMLIB,DISP=SHR
//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
```

4.4.3 START Command Enhancements

A new parameter has been added to the START command: *JOBNAME*. This new parameter allows you to specify a job name for the STC or job that you are starting.

```
START procname.identifier,keyword=option,keyword=option....,
          JOBNAME=jjobname,SUB=subsystemname
```

4.4.3.1 Started Task Processing

When you start a started task, the system determines whether the START command refers to a procedure or a job. First, the system checks for the existence of an IEFJOBS DD within the MSTJCLxx member. If the IEFJOBS DD exists, the system searches the IEFJOBS DD concatenation for the member requested in the START command. If there is no member by that name in the IEFJOBS concatenation, or if the IEFJOBS concatenation does not exist, the system searches the IEFPDSI DD for the member requested in the START command.

If a member is found, the system examines the first record for a valid JOB statement and, if one exists, uses the member as the JCL source for the started task.

If the member does not have a valid JOB statement in its first record, the system assumes that the source JCL is a procedure and creates JCL to invoke the procedure. After JCL source has been created (or found), the system processes the JCL.

4.4.3.2 Security Considerations

Security in the started task environment is managed by RACF by means of a new class. The new class in RACF V2.1 called STARTED enables the security assignments for started tasks to be changed or added dynamically. This improves the use of the started task table, ICHRIN03 used today, which requires an IPL CLPA to update it. However, users must keep an ICHRIN03, or RACF fails to initialize. See *RACF V2 Planning: Installation and Migration* for complete information.

Since started tasks using a job as their source still run as started tasks, the security considerations are the same as with started tasks started using a procedure.

4.5 I/O Path Validation

In pre-MVS/ESA SP 5.1.0 versions, a device could be logically online although no physical paths existed to the device. This situation is confusing, and problem resolution is often time consuming.

To provide better awareness and knowledge of devices and their physical paths, MVS/ESA SP 5.1.0 introduces the following new macro services:

IEEVARYD The IEEVARYD macro service allows a caller to vary one or more devices online or offline on a single system. It has the same effect as the VARY device operator command, but it provides return and reason codes to the calling program, rather than issuing messages to a console.

IOSCDR Allows an authorized application program to retrieve device-descriptor information, such as serial numbers and model numbers, that uniquely identifies I/O hardware located along a specific I/O path. The IOSCDR macro maps the data in the IHACDR. A detailed explanation of the fields can be found in the *MVS/ESA SP V5 Data Areas, Vol 2 (DCCD-ITTCTE)* manual. This information enables a system programmer to:

- Uniquely identify components, such as devices or control units, across multiple systems.
- After device installation and before bringing the device or path online, check device paths to ensure that cables are connected to the proper devices.
- Construct a map of an installation′s configuration.
- During problem diagnosis, ensure that all paths to a given device reach the expected device.

A sample program is provided in SYS1.SAMPLIB member IOSSCDRE.

IOSPTHV The IOSPTHV macro allows an authorized application program to validate the physical connectivity of a channel path to a device and return meaningful information about path-related errors. This information can be used in the initial diagnosis of path related problems. After installation of a device, channel-to-device connectivity can be verified before a device is varied online. In a multisystem environment, it is possible to remove the physical connection to a device that was previously logically and physically online without first varying it offline. With a program that issues the IOSPTHV macro, it is possible to verify these devices and analyze the error information returned by the macro.

4.5.1 Application Use of Macros

These new macros are mainly intended to be used by ESCON Manager V1 R3, to allow it to perform sysplex-wide operations.

Figure 44 shows an ESCON Manager end user or application program using the IEEVARYD macro to specify that device 200 on system A be varied offline throughout the sysplex, which consists of systems A, B, C, D, and E. ESCON Manager distributes the request to all systems to which the device is attached, and causes each system to vary the device offline. The device, known to systems A, B, and C as device 200, is varied offline on those systems. The same device, known to system D as device 300, is also varied offline on System D. No action is performed on system E, as the device is not attached to that system.

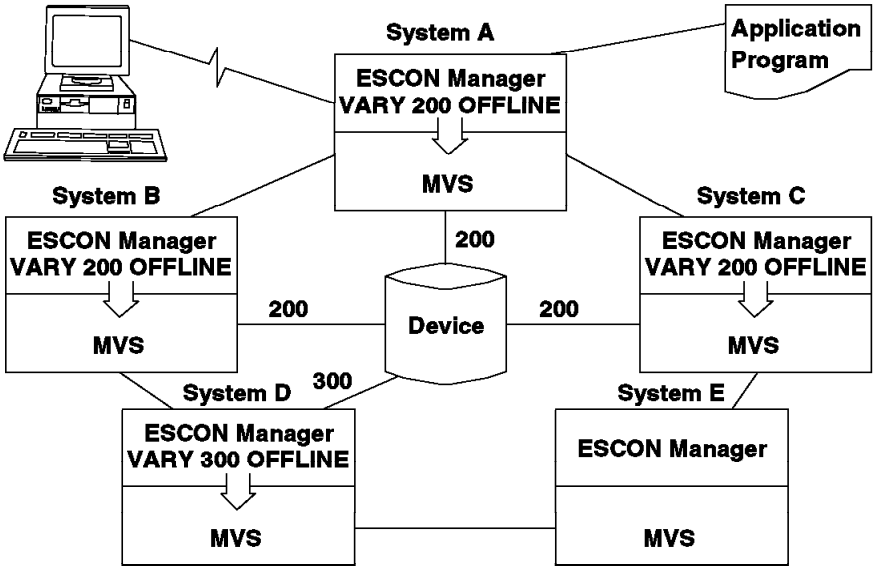


Figure 44. ESCON Manager Use of the New Macros

The IEEVARYD macro provides ESCON Manager with the interface for performing vary operations on a single system.

In order for ESCON Manager to know that device 200 on systems A, B, and C is the same as device 300 on system D, ESCON Manager must obtain I/O configuration data from all systems in the sysplex and correlate the data to identify common resources, such as devices shared by multiple systems. ESCON Manager is able to correlate a shared device in either of the following cases:

- The device supports the device self-description feature. Such a device provides the operating system with a unique identifier. In this case, the device does not need to have the same device number on all systems that share the device.
- The device is defined once in a single I/O Definition File (IODF) that is shared by multiple systems. In this case, the device will have the same device number on all systems that share the device.

Shared devices that do not support the device self-description feature and that are not defined with one definition in the same IODF are treated as distinct devices by ESCON Manager.

4.5.1.1 IOSCDR Macro Use

To perform sysplex-wide operations, ESCON Manager must be able to correlate a physical device with the different device numbers to which the device is assigned on each single system. The IOSCDR service gives ESCON Manager this capability.

Device self-description is the capability of a device to provide data that physically describes the device itself (device descriptors), and specific hardware encountered along the I/O path to the device (control unit descriptors). The descriptors are returned in a configuration data record (CDR) when a read configuration data channel command word is issued to the device. The data contained in the device descriptors can be used to uniquely identify a device in the I/O configuration, as it contains the device type, model, and serial number. CDRs are collected by IOS at I/O validation time (at IPL and VARY ONLINE time), and stored in an internal table. Before MVS/ESA SP 5.1.0, there was no API (macro or service) available to obtain CDRs for a device.

The new IOSCDR macro allows a program to retrieve the CDRs for a device, either from the internal table maintained by IOS, or directly from the device by issuing the RCD CCW.

As shown in Figure 44, each instance of ESCON Manager in the sysplex can invoke IOSCDR, obtain the CDR for the DASD device, and extract the device type, model, and serial number from the CDR. In this way, ESCON Manager knows that the DASD is defined as device number 200 on systems A, B, and C, and is defined as device number 300 on system D.

4.5.1.2 IOSPTHV Macro Use

IOSPTHV is the interface provided by IOS to allow programs to verify channel and device physical connectivity and availability. ESCON manager uses this function to retrieve and display the status of devices. In Appendix A, “A Sample Program Utilizing the IOSPTHV Macro” on page 193, is a sample program that is executed as a TSO/E command processor. This program prompts the user for a device number and returns path related information pertaining to the device.

The new macro IOSPTHV gives a program the means to perform the basic I/O path validation process for a device, that is, verifying the physical availability and connectivity of the device through the channel paths.

IOSPTHV provides the mechanism for ESCON Manager to maintain an accurate status of the devices in the I/O configuration, without having to examine MVS internal control blocks.

4.6 Hardware Configuration Definition Enhancements

In MVS/ESA SP 5.1.0, the MVS Configuration Program (MVSCP) is replaced by the Hardware Configuration Definition (HCD). There are new enhancements in support of MVS/ESA SP 5.2.0 in HCD Version 5 Release 2.

The Hardware Configuration Definition product, as part of MVS/ESA SP 5.2.0, has many new and enhanced functions available, including:

- S/390 microprocessor cluster support
- Coupling facility support
- ESCON Director switch enhancements

- 4-digit device number support
- Startup profile
- Graphical configuration support
- IODF enhancements
- Ease of use changes
- Enhanced migration

HCD 5.2.0 is a new HCD release shipping as a component of MVS/ESA SP 5.2.0. As such it is delivered on the JES2 and JES3 MVS SP product tapes as separately installable FMIDs. HCD 5.2.0 runs on any processor supported by MVS/ESA SP 5.2.0. All devices supported by MVS/ESA SP 5.2.0 can be configured by HCD 5.2.0.

For more information about HCD, see *MVS/ESA SP V5 HCD: User's Guide* .

4.6.1 S/390 Microprocessor Cluster Support

HCD 5.1 provides support for the new S/390 microprocessor cluster having their support elements interconnected by a token-ring LAN. Each support element (SE) of a CPC configured in a cluster has to be connected to the same network as the master cluster, which controls the cluster, as shown in Figure 45.

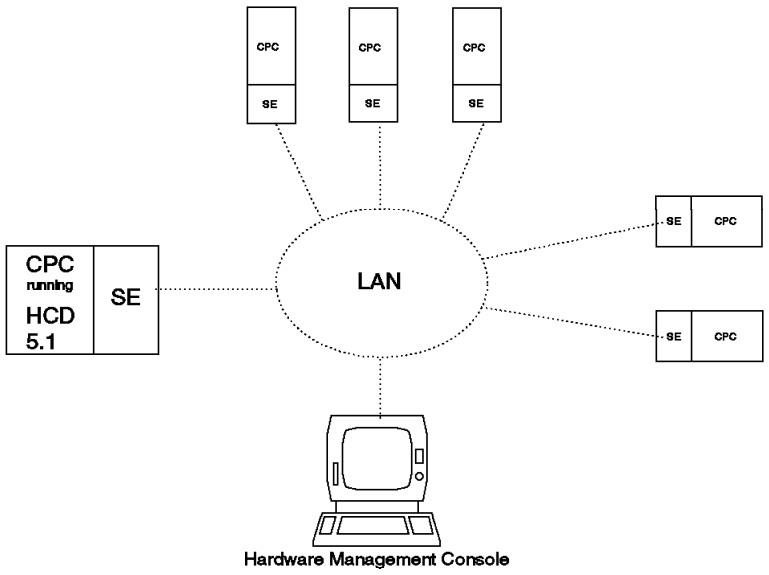


Figure 45. S/390 Microprocessor Complex

HCD, shown running on one of the CPCs, provides a single point of control to perform functions for all the CPCs configured in the same cluster. The functions performed are:

- IOCDS management
- IPL attribute management

That means these functions are available for the local CPC (the processor HCD is running on) as well as for all remote CPCs within the S/390 microprocessor complex. To allow the user to manage all the IOCDSs and IPL attributes within the cluster. This is the interface between HCD and the service processor interface (SPI) of the CPC/SE where HCD is running. Figure 46 on page 99 shows an example of the new HCD panel.

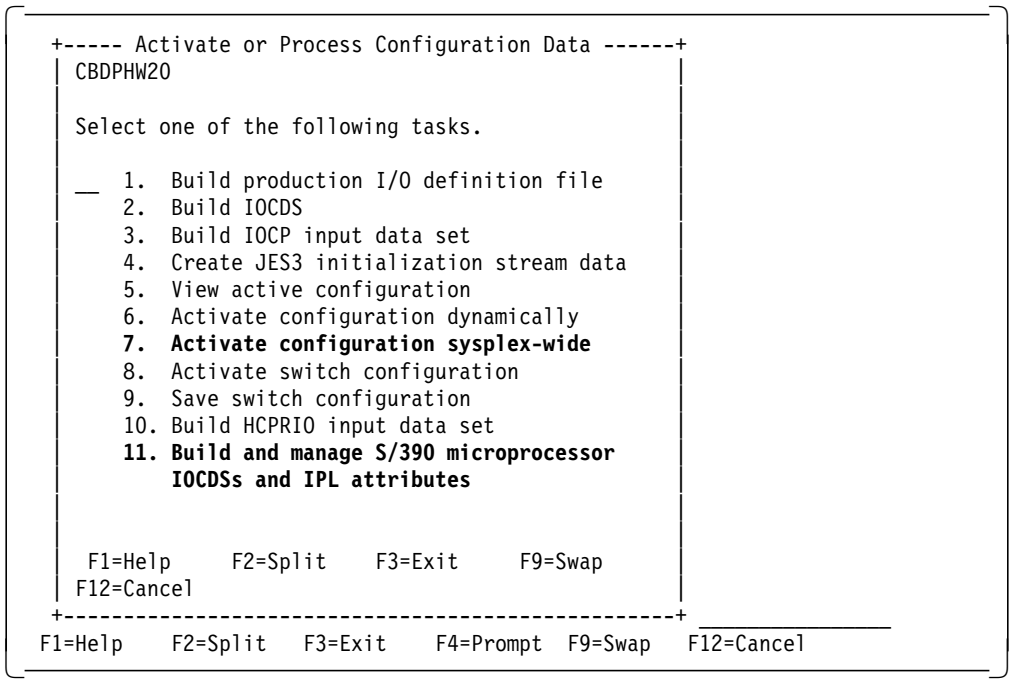


Figure 46. The HCD Activate or Process Configuration Data Panel .

Note: Option 11 has been included for the new S/390 microprocessor cluster support. Option 7 has been added with HCD 5.2.0 to activate IODF sysplex-wide.

HCD introduces a new list displaying all CPCs configured in the same cluster as the CPC on which HCD is running. Each CPC displayed in this active cluster list is identified by the network address (SNA Address) of its support element.

The network address is used by HCD to associate a CPC/SE with a processor definition of the IODF. To enable this relation, a processor has to be defined in the IODF for each CPC configured in the S/390 microprocessor complex. For each processor definition, the SNA address of the support element has to be specified and is saved together with the other processor data in the IODF.

When the active S/390 microprocessor complex list is displayed, the SNA address returned by the hardware management console(HMC) is compared with the SNA address saved in the currently selected IODF. If a match is found, the association is successful, and further functions for the IOCDS and IPL attributes of the CPC can be performed by the user.

4.6.2 Coupling Facility Support

The coupling facility enables high-performance data sharing among processors. Coupling facility capable processors allow the use of a logical partition as a communication partition and high-speed channels to connect the partition with other processors. Support is provided in HCD 5.1 to configure the coupling facility. The coupling facility enables MVS images in a sysplex to communicate and share data with each other. Channel path definitions in the IODF specify the connections from a coupling facility-capable processor to a coupling facility partition.

HCD is enhanced to provide the definition of coupling facilities. HCD provides a mechanism to the user to define all objects that are necessary for coupling

facilities. When you connect a CFS channel path to a CFR channel path, HCD generates the corresponding CFS control unit and devices.

4.6.3 Switch Enhancements

With HCD 5.1, switch configurations can be migrated from an active ESCON Director or an ISPF table. If you have ESCON Manager installed with its capability to read and update ESCON Director files, HCD 5.1 allows you to:

- Save a switch configuration, which you do not want to activate immediately, in an ESCON Director file
- Migrate the saved switch configuration from the ESCON Director file for further processing and manipulating by HCD 5.1 when adding a switch

In previous releases of HCD, the minimum port range was set to “Installed” when you added a switch. You can now set a port range larger than the minimum range to “Installed” when defining a switch.

4.6.4 4-Digit Device Number Support

HCD 5.1 allows you to specify four digits (numbers higher than X′0FFF′) for device numbers and connect these devices to an MVS operating system. HCD 5.1 validation checks whether the specified devices can have 4-digit numbers. This validation is based on the UIMs information. The device numbers for MVS/ESA SP 4.3.0 and prior versions are still restricted to three digits.

4.6.5 Startup Profile

HCD 5.1 provides a profile data set for setting the processing options. For example:

- You can map the control unit types in IOCP input data sets to valid control unit types supported by HCD 5.1. This means that you do not have to change the data sets before migrating them.
- You can select the trace options to be active when using the HCD 5.1 dialog or an HCD 5.1 batch job.
- You can specify defaults for colors and for the layout of the graphical configuration report.

4.6.6 Graphical Configuration Reports

HCD 5.1 allows you to create or view various types of graphical configuration reports of logical configurations in an IODF:

- Reports are stored in DCF format or bookmaster format in a data set for later processing.
- To print the reports, an AFP printer is required.
- To display a report, GDDM and a graphical capable terminal is required. If you are using a programmable workstation and you communicate with the host using a 3270 emulator session, the GDDM-OS/2 link files must be installed on your workstation. You can also save a report as displayed on the screen and print it using the GDDM print facility.

The printed output can be used for documentation purposes and serve as a base for further configuration planning. You can limit the report by means of filtering.

The display function allows you to get a quick overview of your logical hardware configuration. You can use the LOCATE command to find a specific object, and you can jump from an object to the related object list.

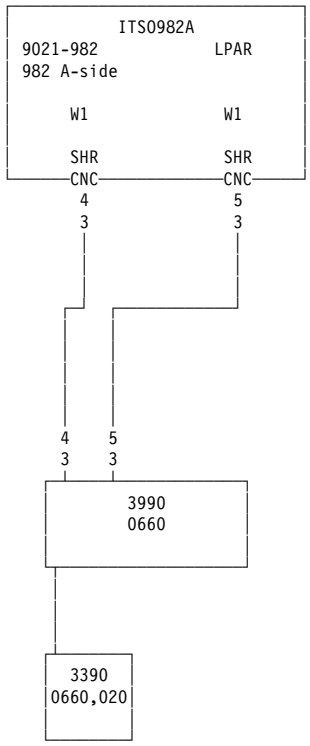


Figure 47. An Example of a Graphical Configuration Report

4.6.7 UCBs Above 16MB

MVS/ESA SP 5.2.0, in conjunction with DFSMS/MVS 1.3, gives you the possibility to define devices to MVS such that the UCB representing the device will reside in 31-bit addressable storage above 16MB. Whether the UCB common segment may reside in the 31-bit storage is controlled by the the new LOCANY device parameter in the HCD I/O device list. See Figure 63 on page 153.

4.6.8 IODF Enhancements

The existing REPEAT pull-down choice of the ADD action has been extended to enable you to copy a selected object together with its dependent objects and I/O attachments not only within the same IODF but also to another IODF.

The target objects may or may not exist in the target IODF. If the objects do not exist in the target IODF, new ones are created by copying the values and definitions of the objects in the source IODF. If some of the objects already exist, a merge action takes place; that is, the definitions of the objects in the source IODF are combined with those of the objects in the target IODF.

You can view information about the currently active IODF and, if available, the configuration token that is currently active in the HSA (hardware system area). You can also see the current activation scope (software changes, hardware changes, or recovery required) and examine the reasons for any limitations to the scope.

You can now transmit IODFs to different MVS levels. For example, a production IODF created with HCD 5.2 can be exported to a MVS 4.3 system and imported there with installed HCD and used for IPL. The IODF format is not changed. The default size of the IODF has been increased to 1024K blocks.

4.6.9 Ease of Use Changes

Up to now an action on an object was initiated either by specifying an action code or by selecting an object, moving the cursor to the action bar and selecting an action. This selection mechanism has been changed to simplify it and be consistent with ISPF 4.1. Select one or more objects from an object list, like the control unit list, and press enter. A context menu pops-up where you can select the required action. Of course, you still can use action codes and prompt for action codes.

The filter support has been enhanced. You can now filter for connections:

- Show all free CHPIDS
- Show all control units that are not connected to a channel path
- Show all ports of a switch that are free

The primary selection panel has been restructured. All options for defining a configuration have been moved to a separate panel showing the hierarchy of the objects you work with.

The GOTO command has been enhanced and allows you to go directly to the list panels of dependent objects or to a specific object within a list. For example, you can go directly to a specific partition or channel path list of a processor. Figure 48 shows an example of the HCD 5.2.0 Primary Selection Panel.

```
                        MVS/ESA HCD Version 5 Release 2

                             Hardware Configuration

Select one of the following.

_   1.  Define, modify, or view configuration data
    2.  Activate or process configuration data
    3.  Print or compare configuration data
    4.  Create or view graphical configuration report
    5.  Migrate configuration data
    6.  Maintain I/O definition files
    7.  Query supported hardware and installed UIMs
    8.  Getting started with this dialog
    9.  What's new in this release

For options 1 to 5, specify the name of the IODF to be used.

I/O definition file . . . 'SYS1.IODF33'                          +

Command ===> ____________________________________________________________
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=Swap      F12=Cancel
```

Figure 48. An Example of the Primary Selection Panel

4.6.10 Removed Restrictions

HCD 5.1 allows you to change the channel path type from parallel to serial (or vice versa), and the operation mode from nonshared to shared (or vice versa). However, changing the type or operation mode of one channel path results in changing all channel paths attached to the logical control unit affected.

The access of candidate list matrix makes it easy to change the lists.

4.6.10.1 SYS1.NUCLEUS Concatenation Removal

In previous releases, HCD requires that SYS1.NUCLEUS is contained in the ISPF load library concatenation. This is because SYS1.NUCLEUS contains all UIMs and associated UDTs.

HCD has been enhanced to no longer require that SYS1.NUCLEUS is in the ISPF load library concatenation nor in the LNKLSTxx member. HCD 5.2 will now automatically allocate SYS1.NUCLEUS at initialization time.

Two new keywords (UIM_LIBNAME and UIM_VOLSER) have been added to the HCD profile. These keywords basically allow the user to specify the name and volume serial number of the library that contains the UIMs. The default name is SYS1.NUCLEUS. The ability to specify the volume serial number is necessary since the library holding the UIMs might not be cataloged.

4.6.10.2 Control Unit Number and Partition Name Change

Up to now, it was not possible to change the control unit number of an already existing control unit in the IODF. HCD requires that the control unit is to be deleted and re-added.

The control unit dialog has been enhanced with HCD 5.2.0 to allow that the control unit number to be over-typed, so another number can be assigned to the control unit. However, this is only allowed via the CHANGE action. The control unit number cannot be changed on the CU list panel since this would result in a change of the “key-column” which is not supported by the generic list handler.

Using the CHANGE action on the partition list, you can change the name of a partition.

4.6.11 Enhanced Migration

You can update existing processor and operating system configuration definitions in an IODF with IOCP, MVSCP, or HCPRIO control statements. In particular, you can:

- Add or replace control units, devices, channel paths, and partitions
- Replace list of consoles
- Update EDTs, generics, and esoterics

The last values entered for the input data sets on the migration panel are retained. You do not have to re-enter the values for a subsequent migration.

Devices defined in the IODF and in the IOCP input data sets need not be connected to exactly the same set of control units. If one set is the subset of the other, the definitions are merged. Previously, the definitions were rejected.

Previously, device numbers could be associated with the wrong processor or partition in the IODF when migrating the MVSCP input data sets if they did not

specify control unit numbers. To resolve this conflict, you are now able to associate an MVSCP input data set with a processor or a logical partition.

An improved mapping algorithm for devices results in fewer duplicate device numbers and in fewer migration failures due to control unit configuration mismatches.

Labels in the input data sets no longer need to be unique. Duplicate labels are ignored.

You can use the startup profile to map the control units to valid HCD control units and reduce the need to change the IOCP input data sets before migrating them.

The migration routine has been changed in HCD 5.2.0 to supply a token value for each esoteric that is coded in the MVSCP deck by means of a UNITNAME statement. The token will be assigned consecutively, starting from 1. However, the token will only be assigned when the appropriate keyword (ESOTERIC_TOKEN=YES) is specified in the HCD profile.

There is no token assigned when performing a “partial migration.” In order to allow the user to supply an esoteric token when doing a “partial migration,” the new keyword TOKEN has been implemented on the UNITNAME statement.

For those who continue to edit their IOCP and MVSCP decks, HCD 5.2 supports new statements and keywords to migrate, for example, switch data or device parameters like DYNAMIC and LOCANY.

4.6.12 Restrictions in HCD

MVS/ESA SP 4.3.0 is the last release that supports MVSCP. This means that an IODF file is required to IPL a MVS/ESA SP Version 5 system. The migration facilities have been enhanced to ease the transition to HCD 5.1. Users that are still running on pre-MVS/ESA SP 4.1.0 releases and that are migrating to MVS/ESA SP 5.1.0, can overcome this restriction by using HCD at installation time to create the IODF file.

HCD 5.1 requires a security product, such as RACF, before IOCDS management can take place. If a security product is not installed, then HCD does not allow access to option 2.10, “Build and manage S/390 microprocessor IOCDSs and IPL attributes.” When attempting to write an IOCDS using option 2.2, the operator is prompted for confirmation of the IOCDS write operation. Additionally, the MVS ACTIVATE command only works with a security product.

Please refer to *MVS/ESA SP V5 HCD: User's Guide* for more details on security requirements.

4.6.13 Maintaining Esoteric Order

In the past, you may have had problems related to the IODF versus MVSCP incompatibility with esoteric ordering. This is because when IPLing and using an IODF, the EDT is built in esoteric name sequence (alphanumeric, ascending). You do not have control over the order of the esoterics within the EDTs. Cataloging by esoterics has been discouraged for some time now. The reason is that part of what makes up the four byte entry within the catalog is a two byte relative index of the esoteric within the eligible device table (EDT). That works

as long as you don't move esoterics around (or insert a new esoteric) within the MVSCP definition.

It has always been pointed out in the documentation that you should not catalog by esoteric. However, many customers have old applications and JCL that catalog by esoteric and it is almost impossible for them to be converted.

HCD introduces a "token" that is associated with an esoteric. You can specify the token whenever you define or change the properties for an esoteric. The token is an arbitrary decimal number (1 to 999) that corresponds to the esoteric index used within the catalog. The specification of the token is optional, but when a token is specified for at least one esoteric it must be specified for all other esoterics as well.

The token is used by allocation to find the proper esoteric for a data set that has been cataloged using the esoteric.

4.7 4-Digit Device Support

MVS/ESA SP 5.1.0 enables installations to use four hexadecimal digits when defining device numbers. This theoretically increases the maximum number of devices that can be defined to a single MVS image from 4,096 to 65,536. However, if an installation wants to limit the amount of virtual storage below 16MB that is used for devices to the amount used in the previous release, it must limit the number of additional devices defined to approximately 1500.

MVS/ESA SP 5.2.0 with the UCB virtual storage constraint relief (VSCR) enables you to move the UCBs above 16MB. This will improve systems management of the IODF in a SYSPLEX environment. So you may define up to the limit of 65,536 devices without being constrained by limits to below 16MB virtual storage.

For more information about UCB VSCR, refer to Chapter 8, "UCB Virtual Storage Constraint Relief" on page 139.

4-digit device specification makes it easier for installations to use the same I/O definition files (IODFs) for multiple systems and provides more flexibility for unique device numbers across a sysplex. It is unlikely that users will use the full compliment of device numbers. However, the possibility now exists for an installation to have a more varied range of device numbers. By defining the I/O subsystem in such a way that the types of devices are assigned to unique groups of device numbers, it makes it easier to manage large installations. An example would be to assign all CTCs to a device number range of 4000-4FFF.

The possibility now exists for device numbers to be equal to unit types. An installation can, for example, have a device type of 3380 installed and also have a device number of 3380 assigned to a printer. To differentiate between a generic unit type and the device number when coding JCL, a / has to be coded on the UNIT= statement ahead of the device number. For devices up to numbers X'FFF', the / is optional. All other device numbers have to have a leading /. The following example shows the different combinations of coding JCL using 4-digit device numbers.

To allocate device number 660:

```
//SYSUT1  DD  UNIT=660     /* Device number 660            */
//SYSUT1  DD  UNIT=/660    /* Device number 660            */
//SYSUT1  DD  UNIT=/0660   /* Device number 660            */

//SYSUT1  DD  UNIT=3380    /* Unit is a 3380 DASD device   */
//SYSUT1  DD  UNIT=/3380   /* Device number 3380 a printer */
```

To allocate device number 313F, a / has to be coded:

```
//SYSUT1  DD  UNIT=/313F   /* Device number 313F           */
```

Each application used in an installation has to be evaluated and tested for its dependency on 3-digit device numbers.

If an IODF containing 4-digit devices is used to IPL a MVS 4.3 system, the 4-digit devices are ignored. That means, no UCBs are built for the 4-digit devices.

4.7.1 UCB Services

Prior to MVS/ESA SP 5.1.0, the UCB prefix, common segment and common extension were all located below 16MB. With Version MVS/ESA SP 5.1.0, the UCB prefix has been moved above 16MB into ESQA.

Shared pages and UCB VSCR, introduced in MVS/ESA SP 5.2.0, allows all of the UCB to reside above 16MB. A new programming authorized interface IOSCAPU is introduced that supports CAPTURE, UNCAPTURE and TRANSLATE from captured to actual UCB addresses. A fast path authorized service called IOSCAPF is also available to translate a captured UCB address to an actual UCB address. The UCBEXTP and UCBEXTPT fields in the UCB are not valid for above 16MB and captured UCBs.

In support of 4-digit device support, the following UCB services have been updated.

- UCBLOOK
- UCBSCAN
- UCBINFO

UCBSCAN and UCBLOOK have a new LOC keyword to protect programs that are sensitive to addressability. The different options on the LOC keyword are as follows:

- BELOW - The caller is to receive addresses of below 16MB UCBs only. This is the default when the caller does not specify PIN as well.
- ANY - The caller is to receive addresses of UCBs regardless of residency.

UCBSCAN can also return the address of the UCB common extension by using the UCBCXPTR parameter, and the UCB prefix extension by using the UCBPXPTR parameter. Use of these services is the recommended approach for obtaining any information from the UCB.

The use of IOSLOOK and IOSINFO is still supported. However, they do have restrictions on the UCBs they can access. In addition there are new macros, IOSCMXA and IOSUPFA, to find the address of common and prefix extensions of the UCB. IOSCMXA and IOSUPFA were introduced in MVS/ESA SP 5.1.0 and are still supported in MVS/ESA SP 5.2.0. Improved performance of authorized

services for finding the addresses of the UCB common extension and the UCB prefix extension segment are provided by IOSCMXR and IOSUPFR macros. These new macros, introduced in MVS/ESA SP 5.2.0, handle parameter passing in general purpose register (GPR) 1. For details, see *MVS/ESA SP V5 Auth Assembler Services Reference ENF-ITT*.

Refer to *MVS/ESA SP V5 Auth Assembler Services Guide* for a description of these changes and assistance in their use.

4.8 Display RESERVE Status

When a start pending missing interrupt (MIH) condition occurs in a multisystem environment (including non-MVS systems), IOS first does a D U,VOL=volser to get the correct device number from the target system. Once the device number is determined, MVS automatically displays the central processor serial number of a failed system and the device address of the device holding the reserve. If the failed system is in a sysplex, the system name is included in the display. From this information, the Input Output Supervisor (IOS) internally routes a more specific command, D GRS,DEV=(xxxx), to the system holding the resource. The display that follows enables an installation to determine what job or address space is holding the resource. This allows the installation to distinguish between MIH conditions that cause long-held reserves and other more complex configuration problems.

Using the information provided in the display reserve status message (IOS431I) and the accompanying ISG020I message, operators and system programmers can free the resource and reduce the duration of system outages. The display reserve status message (IOS431I) is issued automatically.

In a sysplex environment, *the couple data set has to be formatted at* the MVS/ESA SP 5.1.0 level. The couple data set at this level has additional fields that contain information regarding each system that joins the sysplex. This information is used to identify the system in the process of displaying reserve status.

In Figure 49 on page 108, system SC52 is holding an exclusive reserve on volume CKPT1, unit address(204). Job POLUPD running on System SC53 wants to get at the same resource, and the system displays START PENDING messages. The MIH detects the reserve condition and issues message IOS431I. The IOS has identified the device address that is causing the RESERVE problem and issues an internal command D GRS,DEV=(204), which is not displayed on the console. A message is then issued identifying the jobname and system name that is holding the resource. In the example, it is jobname ANDREVN, ASID 0022 on system SC52, that is causing the reserve. The operator can decide if it is important enough to cancel the job or let it continue until completed.

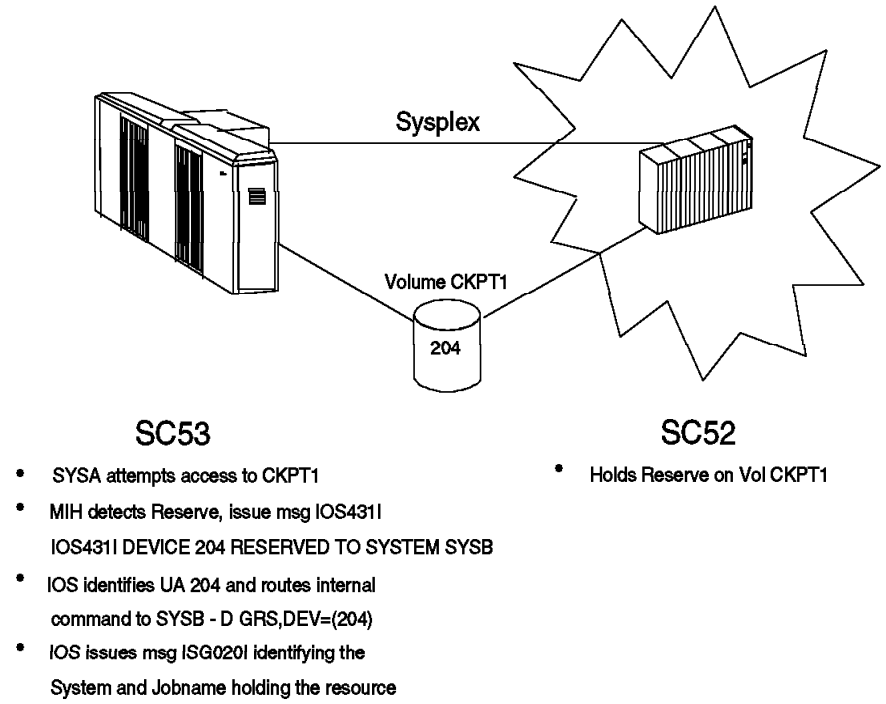


Figure 49. Display RESERVE Status

An example of the SYSLOG is as follows:

```
IEA631I  OPERATOR SYSIOSRS NOW INACTIVE, SYSTEM=SC53    , LU=IOSAS
IEF196I IOS071I 0204,**,POLUPD, START PENDING
IOS071I 0204,**,POLUPD, START PENDING 326
IEF196I IOS071I 0204,**,*MASTER*, START PENDING
IOS071I 0204,**,*MASTER*, START PENDING 328
IOS431I DEVICE 0204 RESERVED TO CPU=0102569672,LPAR ID=1 330
        SYSTEM=SC52
IEA630I  OPERATOR SYSIOSRS NOW ACTIVE,   SYSTEM=SC53    , LU=IOSAS
 ISG020I 13.26.20 GRS STATUS 452 332
 DEVICE:0204 VOLUME: CKPT1 RESERVED BY SYSTEM SC52
 S=SYSTEMS EXECENQ  EXECEXEC
 SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
 SC52      ANDREVN           0022       009B3CF0 EXCLUSIVE   OWN
```

4.9 Dynamic Exits

Before MVS/ESA SP 5.1.0, installation exit modules were hard coded. This made it very difficult, if not impossible, to change exit modules and implement the change while the system was up and running.

Additionally, you could only have one module associated with an exit point. This meant that if many different products needed to use the same exit, it was

necessary to provide a single module that fulfilled all the requirements of those different products.

Finally, and most crucially, you could only implement any change you made to the exit module by means of an IPL. This, of course, meant outages to your system.

MVS/ESA Version 5 Release 1, together with DFSMS/MVS 1.1 or higher, introduces the dynamic exits capability. This new facility provides system control of multiple exit routines called for an exit, and (in some cases) allows an installation to update or refresh an exit without the need to re-IPL the system.

In MVS/ESA V5.1, the IBM SMF and allocation exits exploit this new capability. Users can associate multiple exit routines with the SMF and allocation exits and control their use at IPL or while the system is running.

The dynamic exits function allows the following:

- Updates to exit modules without IPL
- Multiple modules per exit point
- Improved availability and usability

The exits can be updated in various ways. They may be added singly by means of the changed SETPROG operator command. Or if you have many exit changes to implement, they may be brought in at IPL time in two ways:

- By replying with PROG=xx to the following message:

  ```
  IEA101A SPECIFY SYSTEM PARAMETERS FOR RELEASE xx.yy-VER=ww...w
  ```

- The operator can specify PROG=xx once system commands are accepted.

Note: Dynamic exits can also be used by your own installation-written exits.

The following list shows the IBM supplied exits that are automatically defined to the dynamic exits facility.

- SYS.IEFACTRT -- SMF Job/Step Termination Exit
- SYS.IEFUAV -- User Account Validation Exit
- SYS.IEFUJI -- Job Initiation Exit
- SYS.IEFUJP -- Job Purge Exit
- SYS.IEFUJV -- Job Validation Exit
- SYS.IEFUSI -- Step Initiation Exit
- SYS.IEFUSO -- SYSOUT Limit Exit
- SYS.IEFUTL-- Time Limit Exit
- SYS.IEFU29 -- SMF Dump Exit
- SYS.IEFU83 -- SMF 83 Record Exit
- SYS.IEFU84 -- SMF 84 Record Exit
- SYS.IEFU85 -- SMF 85 Record Exit
- SYSSTC.IEFUJP -- Subsystem/Started Task Job Purge Exit
- SYSSTC.IEFUSO -- Subsystem/Started Task SYSOUT Limit Exit
- SYSSTC.IEFU29 -- Subsystem/Started Task SMF Dump Exit
- SYSSTC.IEFU83 -- Subsystem/Started Task SMF 83 record Exit
- SYSSTC.IEFU84 -- Subsystem/Started Task SMF 84 record Exit
- SYSSTC.IEFU85 -- Subsystem/Started Task SMF 85 record Exit
- IEF_ALLOC_OFFLN -- Allocated/Offline Device Installation Exit
- IEF_SPEC_WAIT -- Specific Waits Installation Exit
- IEF_VOLUME_ENQ -- Volume ENQ Installation Exit

- IEF_VOLUME_MNT -- Volume Mount Installation Exit
- IEFDB401 -- Allocation Input Validation Routine
- IEASDUMP.GLOBAL
- IEASDUMP.LOCAL
- IEASDUMP.QUERY

Note: The dynamic exits facility uses the *full* name as supplied by IBM for all supported exits.

When calling the exit modules, two choices are available: full-function path and fast path, where:

full-function Dynamic exits processing provides MVS recovery if the exit module abends. Dynamic exits processing runs in system state and calls the exit module in the same state in which the call to the dynamic exits processing was made.

fast Dynamic exits processing provides no recovery in case of an abend. The caller must establish any recovery, and if an abend occurs in an exit, the caller must reinvoke dynamic exits processing with the RECOVER option. Fast path processing runs in the state of the caller.

The dynamic exits facility is implemented using the following functions of MVS:

- A new macro CSVDYNEX
- Multiple exit modules possible for an exit
- Modules defined by parmlib or operator command

For further information, please refer to *MVS/ESA SP V5 Installation Exits.*

4.9.1 CSVDYNEX Macro

The CSVDYNEX macro allows an installation to define exits, associate exit routines with those exits, and control the use of exits and exit routines within a program. Programs may use the new macro CSVDYNEX to exploit the dynamic exits facility. The following functions may be performed with this macro:

- Define an exit
- Add a module to an exit
- Modify the state of an exit module
- Delete a module from an exit module
- Undefine an exit module
- Use the attribute function to change the attributes of an exit
- Call the modules for a particular exit
- Query if there are any modules for a particular exit
- List the exits, or the modules associated with an exit
- Recover from an error in a fastpath call

Authorization to use this macro is through any of the following:

- System authorization facility (SAF)
- The program state
- The program PSW key
- An APF-authorized program.

The RACF FACILITY class profiles may be used to authorizer the caller. The profile to be checked depends upon the type of function being requested.

4.9.2 Parmlib Enhancements

SYS1.PARMLIB member PROGxx has been enhanced to support the new dynamic exits facility. You can use this member to associate exit modules with exit names. The relevant PROGxx member is referenced by means of either a reply to the IEA101A message at system initialization, or by the SET PROG= operator command once the system is accepting commands after an IPL. The IPL options can point to the PROGxx member.

Before MVS/ESA SP 5.1.0, the PROGxx member could be used for specifying APF definitions (the Dynamic APF support was added by MVS/ESA SP Version 4 Release 3). You can specify multiple PROGxx parmlib members on the SET PROG operator command or on the PROG= reply. Therefore for ease of use and maintainability, it is recommended that you keep separate the members containing APF definitions and those containing exit module and exit name definitions.

The keywords and abbreviations you use in PROGxx (after the EXIT word) are the same as those used on the SETPROG EXIT operator command. Whichever method you use for working with exits and exit modules, the result is the same.

The new EXIT statement of the PROGxx parmlib member allows an installation to:

- Add exit routines to an exit that has been defined to the dynamic exits facility
- Modify or delete exit routines for an exit
- Change the attributes of an exit at or after IPL
- Undefine an implicitly defined exit

4.9.2.1 Parmlib Options

The complete list of options for the EXIT statement are:

```
EXIT ADD
     EXITNAME(ex)
     MODNAME(mmmm)
     (STATE({ACTIVE|INACTIVE}))
     (DSNAME(dd))
     (JOBNAME(jjj|*))
     (ABENDNUM(n(,CONSEC)))

EXIT MODIFY
     EXITNAME(ex)
     MODNAME(mmmm)
     (STATE({ACTIVE|INACTIVE}))
     (JOBNAME(jjj|*))

EXIT DELETE
     EXITNAME(ex)
     MODNAME(mmmm)
     (FORCE({YES|NO}))

EXIT UNDEFINE
     EXITNAME(ex)
```

```
EXIT ATTRIB
     EXITNAME(ex)
     KEEPRC(compare,kk)
```

4.9.2.2 Parmlib Exit Examples

The following examples explain the use of the syntax for a PROGxx parmlib member:

1. Add an exit routine, R1, to exit SYS.IEFUJI.

```
EXIT ADD
     EXITNAME(SYS.IEFUJI)
     MODNAME(R1)
     DSNAME(MY.LOCAL.EXITS)
```

2. Modifiy exit routine, R2, for exit SYS.IEFUSI to make it inactive.

```
EXIT MODIFY
     EXITNAME(SYS.IEFUSI)
     MODNAME(R2)
     STATE(INACTIVE)
```

3. Delete exit routine, R3, from exit SYS.IEFACTRT.

```
EXIT DELETE
     EXITNAME(SYS.IEFACTRT)
     MODNAME(R3)
```

4. Define exit SYS.IEFUTL so that a return code 4 produced by this exit is returned to the caller of the exit.

```
EXIT ATTRIB
     EXITNAME(SYS.IEFUTL)
     KEEPRC(EQ,4)
```

We recommend that you convert any EXIT statements currently defined in the EXITxx parmlib member to equivalent statements in a PROGxx member.

You may already have a PROGxx member of SYS1.PARMLIB that is used to define APF authorized libraries, and this may be referred to in your IEASYSxx member. We recommend that you use a separate PROGxx member for your installation exits, and that this member is *not* referred to in your IEASYSxx member. It may be beneficial to have your installation written exits in a different APF authorized library to those supplied by IBM.

The dynamic exits facility uses DFSMS/MVS to dynamically activate and deactivate the libraries containing the installation exit modules. Since DFSMS is not active during the parmlib processing stage of the system IPL, any dynamic exit calls fail. It is therefore recommended to activate the PROGxx member containing your exit requirements at a later stage of the IPL. This can be achieved through the COMMNDxx member, an automation utility or manually. During this project we used a member named PROGEX.

To convert EXITxx to PROGEX, you can use the supplied IEFEXPR REXX EXEC found in SYS1.SAMPLIB.

Refer to *MVS/ESA Initialization and Tuning Reference* for more details on how to specify the PROGxx member in SYS1.PARMLIB.

4.9.3 Operator Commands

The use of dynamic exits and exit routines may also be controlled through the use of MVS operator commands. These commands are:

SET PROG=EX Specify the particular PROGEX parmlib member the system is to use.

SETPROG EXIT Use to add exit routines to an exit, change the state of an exit routine, delete an exit routine from an exit, undefine an implicitly defined exit, and change the attributes of an exit. The SETPROG EXIT command has same actions as SET PROG= or PROG=.

D PROG,EXIT This DISPLAY command displays exits that have been defined or have had exit routines associated with them.

The following example shows a command to add an exit routine, called *UAVEXIT*, and associate this new routine with the SMF exit IEFUAV. The UAVEXIT module can be found in the data set called *LOCAL.SMF.EXITS*, and the desired state of the exit routine is *active*.

```
SETPROG EXIT,ADD,EXITNAME=SYS.IEFUJI,MODNAME=UAVEXIT,
   DSNAME=LOCAL.SMF.EXITS,STATE=ACTIVE
```

Note: Since exits can be removed by this, as an added precaution, an installation may wish to restrict the use of the SETPROG command.

4.9.3.1 Dynamic Exits Display Information

To enable dynamic exit information to be queried and displayed at a system console, or via SDSF, you can use the MVS command:

```
D PROG,EXIT,{{EXITNAME|EX|EN}=exitname  }
            {{EXITNAME|EX|EN}=exitname* }
            {{MODNAME|MOD}=modname      }
            {(ALL)(,IMPLICIT|,IMP)      }
```

The resulting display shows all exits defined to the dynamic exits facility by their listed IBM names. The following example displays information for the SYS.IEFACTRT exit:

```
D PROG EXIT,EX=SYS.IEFACTRT

CSV461I 16.47.57 PROG,EXIT DISPLAY 150
EXIT             MODULE   STATE MODULE   STATE MODULE   STATE
SYS.IEFACTRT     IEFACTRT   A   IEFACTR2   A   IEFACTR3   A
                 IEFACTR4   A
```

4.9.4 Dynamic Exits Considerations

The dynamic exit facility uses DFSMS/MVS to access non-system libraries in order to load your installation exit modules. Since DFSMS is not active at the time of parmlib processing, some of the dynamic functions are not available at this time. This and other restrictions are explained in the following list:

- The exit routines must reside in an APF authorized library.

  This is a requirement of the dynamic exits facility.

- At IPL time you may use only the ATTRIB option.

This is because DFSMS needs to be functioning to allow the loading of your exit routines, and at this stage of the IPL process, DFSMS may not have completed initialization.

- The IBM supplied dummy SMF exits are automatically activated.

  SMF searches SYS1.LPALIB, your MLPA concatenation and then all the linklisted libraries for modules with names matching the exit names; for example, IEFACTRT, IEFU83, IEFU84, and IEFU85. Any modules found are loaded and activated by the dynamic exits facility. No messages appear to indicate the results.

  Note: The reentrancy requirement for SMF and allocation exits is enforced. SMF exits IEFACTRT, IEFU83 and IEFU84 are changed to support only AMODE 31 exit routines. For exit IEFACTRT, general purpose register 1 no longer returns a value indicating that the termination record is not to be written to the SMF data set.

- If you wish to use the same name as the IBM supplied dummy exit, that is to say, IEFACTRT, you must either delete the dummy exit from the system libraries, or use a DELETE with an ADD combination in your PROGxx member.

  For example, there is a dummy module named IEFACTRT in a linklisted library and you wish to activate a test version with the same name. Code your PROGEX member as follows:

  ```
  EXIT DELETE EX(SYS.IEFACTRT) MODNAME(IEFACTRT)

  followed by

  EXIT ADD EX(SYS.IEFACTRT) MODNAME(IEFACTRT) DSNAME(MY.LOCAL.EXITS)
  ```

- The SET SMF=xx command overrides the PROGEX options.

  If a set SMF=XX command is entered and the new SMFxx member does not include the SMF exit names, then the exits are disabled. At this time the DISPLAY PROG command may still show the previous association with the SMF exits. If SMF is now set back to including the SMF exits, SMF reads the dynamic exits association and attempts to re-establish the previous exit associations. However, if the exit module you require was called from a non-system library, dynamic exits does not now know where to get the module from and an error message appears.

  The new exit module will therefore have to be reloaded by the SETPROG or SET PROG= commands.

The following SMF exits have been changed to exploit the new dynamic exits facility:

- IEFACTRT
- IEFUAV
- IEFUJI
- IEFUJP
- IEFUJV
- IEFUSI
- IEFUSO
- IEFUTL
- IEFU29
- IEFU83
- IEFU84

- IEFU85

Four of these exits have been changed to 31-bit mode only (AMODE=31):

- IEFACTRT
- IEFU83
- IEFU84
- IEFU85

Note: You should ensure that these exits have the correct AMODE setting; otherwise the system rejects them.

Chapter 5. MVS Subspaces

Within an application server address space, many application programs run under a single server program. An error in one of these application programs can cause it to overwrite the code or data of the other application programs or of the server program itself. Subspaces provide a means of limiting the application server address space storage that an application program can reference, thus limiting the damage an application program error can do within the application server address space.

This chapter describes the concept of subspaces, when to use them, how to create them, how to manage them, and how to delete them. It also describes considerations for providing recovery for and diagnosing errors in programs that run in subspaces.

5.1 What Is a Subspace?

A subspace is a specific range of storage in the private area of an address space, designed to limit the storage a program can reference.

A program that is associated with a subspace cannot reference any of the private area storage outside of the subspace storage range; the storage is protected from the program. Whether a given range of private area storage is protected from a program associated with a subspace depends on whether the storage is:

- Eligible to be assigned to a subspace (or “subspace-eligible”)
- Assigned to a subspace
- Not eligible to be assigned to a subspace

You control these storage “states” through the IARSUBSP macro. Storage outside of the private area is not affected by subspaces.

A program running in an address space can reference all of the storage associated with that address space. In this chapter, a program′s ability to reference all of the storage associated with an address space is called full address space addressability. A program running with full address space addressability can reference storage in any of the three states: eligible to be assigned to a subspace, assigned to a subspace, or not eligible to be assigned to a subspace.

A program that runs in an address space that owns subspaces also has full address space addressability until it issues an instruction to limit the storage it can reference. In an address space that owns subspaces, issuing the BSG instruction (Branch in Subspace Group, see *ESA/390 Principles of Operation*) controls whether a program runs with full or limited address space addressability. In this chapter, a program running with limited addressability is said to be running in a subspace.

A program running in a subspace can reference storage that is assigned to its own subspace and storage that is not eligible to be assigned to a subspace. It cannot reference storage that is eligible to be assigned to a subspace or storage that is assigned to a subspace other than the one in which the program is running.

In other words, a subspace allows a program running in it to reference all of the storage associated with the address space except the private area storage that is eligible to be assigned to a subspace or storage that is assigned to another subspace.

When storage is not eligible to be assigned to a subspace and not assigned to a subspace, it can be referenced by a program running in a subspace or a program running with full address space addressability. This storage can be referenced by all subspaces as well as by programs running with full address space addressability.

An address space that owns subspaces is also called a “base space.” Figure 50 illustrates the concept of creating a subspace in base space ASID 23.

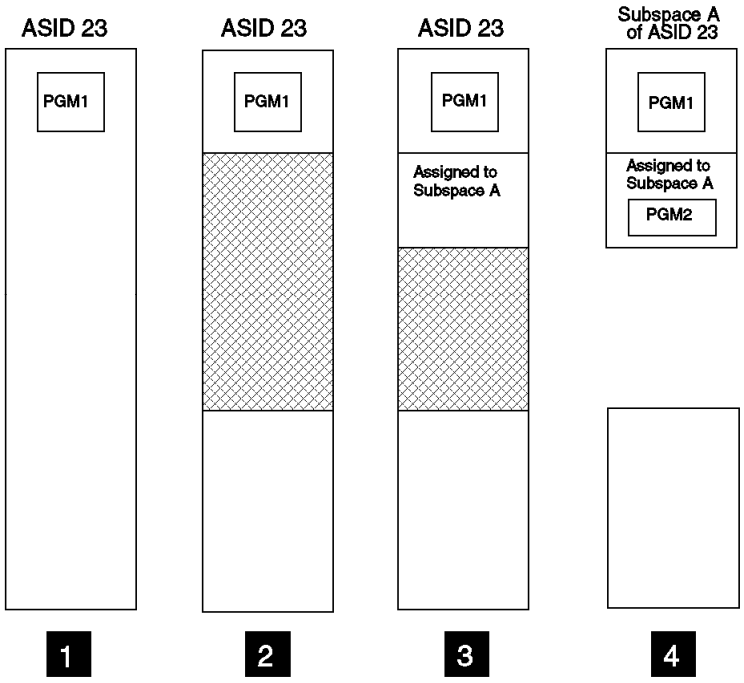


Figure 50. Illustration of an Address Space that Owns One Subspace

1 PGM1 is a program running with full address space addressability in address space ASID 23. ASID 23 owns no subspaces, and no storage eligible to be assigned to a subspace. PGM1 can reference all storage in the address space.

2 PGM1 makes the shaded area of storage eligible to be assigned to a subspace. The eligible storage has not been assigned to a subspace. PGM1 can reference the subspace-eligible storage because PGM1 is not running in a subspace.

3 PGM1 assigns part of the subspace-eligible storage to Subspace A. PGM1 can reference the subspace storage as well as the subspace-eligible storage because PGM1 is not running in a subspace.

4 PGM1 issues the BSG instruction, which passes control to PGM2 to run in Subspace A. PGM2 can reference the storage that is assigned to

subspace A, and storage in the address space that has not been made subspace-eligible. PGM2 cannot reference the subspace-eligible storage while PGM2 is running in the subspace.

An address space can have many subspaces. Each application program running simultaneously in an address space can run in its own subspace. The subspace restricts a program running in it from referencing the storage assigned to other subspaces. Figure 51 illustrates the concept of multiple subspaces by adding another subspace to address space ASID 23.

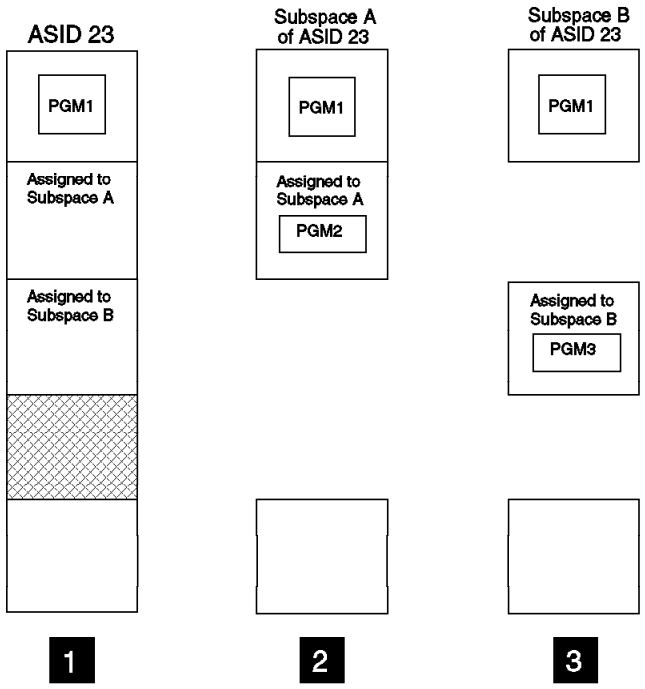


Figure 51. Illustration of Address Space that Owns Two Subspaces

1 Running with full address space addressability, PGM1 creates and assigns storage to Subspace B. PGM1 can reference the entire address space, including storage assigned to Subspaces A and B, and subspace-eligible storage (shaded).

2 PGM2, running in Subspace A, can reference storage that is assigned to Subspace A and storage that has not been made subspace-eligible. PGM2 cannot reference storage in Subspace B or storage that is subspace-eligible.

3 PGM3 is a program running in Subspace B. PGM3 can reference storage that is assigned to Subspace B and storage in the address space that has not been made subspace-eligible. PGM3 cannot reference storage that is assigned to Subspace A, or storage that is subspace-eligible.

The number of subspaces per address space is limited by the amount of unallocated private storage available in the address space, and by the amount of storage assigned to each subspace.

A subspace is associated with only one address space and is owned by the task that creates it. A task cannot pass addressability to its subspaces to its subtasks or SRBs. An attached subtask or an SRB gets control with full address space addressability.

A subspace has an access list entry (called an “entry” in this chapter) associated with it. After a program creates a subspace, it adds the entry to the dispatchable unit access list (DU-AL) associated with the task the program runs under. A program does not have to be in AR mode to use a subspace, although it can be.

A program can toggle between running in a subspace and running with full address space addressability by issuing the BSG instruction.

5.2 Deciding Whether Your Program Should Run in a Subspace

Subspaces are beneficial in an application server address space in which numerous applications run under a single task within the address space.

Using subspaces as described here requires few or no changes to the application programs. However, using subspaces does require additional code in the server program. Further details than are presented here can be found in *MVS/ESA SP V5 Extended Addressability Guide*.

5.2.1 Benefits of Subspaces

The use of subspaces can protect the server and application programs in an address space. In addition, subspaces can help you to identify where in the address space an error has occurred. Subspaces can be used for:

- Protecting the server program: using subspaces in an application server address space protects the server program and its data. Subspaces reduce the number of failures in the server program by protecting it from the errors of other programs in the address space.
- Protecting the application program: using subspaces in an application server address space also protects the applications, similar to the way programs are protected by running in separate address spaces. By preventing applications from overwriting each other’s code and data, subspaces increase the reliability of these applications.
- Providing diagnosis: an IPCS diagnostic report and trace functions can help you to identify where in the address space an error has occurred.

  An ABEND dump can help you to identify that an error resulted from a prohibited storage reference. When requested by a program running in a subspace, an ABEND dump contains only the storage that the program is allowed to reference.

5.2.2 Limitations of Subspaces

Subspaces have the following limitations:

- Subspaces are available on the following IBM hardware platforms:
  - ES/9000 9021 711-based processors
  - ES/9000 9121 511-based processors
  - ES/9000 9221 211-based processors

It is possible for a program to check for the availability of the subspace group facility. To do this you need to check the CVTSUBSP bit in the CVT. When the bit is on, the facility is available.

- Subspaces do not provide protection against deliberate attempts to overwrite code.
- To ensure that subspace storage is protected, the system abnormally ends a program that:
  - Attempts to reference storage to which it does not have addressability
  - Provides incorrect information on the IARSUBSP macro

  Therefore, you might need to code additional recovery routines for your programs.
- An unauthorized application program running in a subspace cannot add more storage to the subspace. If an application program requires more subspace storage, the server program must obtain subspace-eligible storage for the application. This is explained in detail in *MVS/ESA SP V5 Extended Addressability Guide*.

5.2.3 System Storage Requirements

One factor that might influence your decision to use subspaces is the amount of virtual and central storage that the system requires to manage them. This storage overhead can affect system performance.

The system allocates storage for its own use from subpool 255 when a program:

- Makes storage eligible to be assigned to a subspace
- Creates a subspace
- Assigns more than 2 segments of storage below 16 megabytes to a subspace

The system deallocates its storage when the program deletes the subspace or makes the storage ineligible to be assigned to a subspace.

The amount of storage that the system requires for its own use depends on whether the subspace storage is above or below 16 megabytes. The system requires more storage to manage subspaces below 16 megabytes. Use the following guidelines to plan for the system's storage requirements:

Table 4. System Storage Requirements When Managing Subspaces

The System Uses:	**For Each:**
8192 bytes	Address space in which a program issues IARSUBSP IDENTIFY
1024 bytes	Segment below 16 megabytes specified on IARSUBSP IDENTIFY
10376 bytes	Subspace created
1024 bytes	Segment below 16 megabytes specified on IARSUBSP ASSIGN, after the first two such segments

5.3 Updating the Application Server to Use Subspaces

Most application servers consist of at least two types of programs:

- Application programs, which perform the work
- A server program, which manages the application programs and the address space

You can choose to manage subspaces in either of the following ways, or with a combination of the two:

- Create a number of subspaces prior to receiving requests for application program services.
- Create one subspace at a time, in response to receiving a request for application program services.

The method that you choose depends on whether your installation is more concerned with storage constraints or performance of the application server.

5.3.1 Managing Subspaces When Performance Is a Priority

It is most efficient to obtain storage for and create the number of subspaces needed for all application programs as part of application server initialization. Then, as a request for an application program′s services is received, the server program assigns eligible storage to a subspace, runs the application program in the subspace, and disassociates the eligible storage from the subspace. As part of application server termination, the application server deletes the subspaces and makes the storage ineligible to be assigned to a subspace.

Managing subspaces in this way is less costly than other designs in terms of performance. The IDENTIFY and CREATE functions (of the IARSUBSP macro) use more instructions than the ASSIGN and UNASSIGN functions. By reserving a quantity of subspace-eligible storage and creating subspaces that are reused for multiple invocations of the application programs, the server program manages subspaces efficiently.

This design could cause storage constraints. When storage is subspace-eligible but not assigned to a subspace, a program running in a subspace cannot reference it. Subspace-eligible storage cannot be released until the server program makes it ineligible to be assigned to a subspace. Furthermore, a program cannot pass ownership of subspace-eligible storage to a subtask.

If storage constraints in the application server address space are a concern at your installation, you might want to consider the alternate design described next.

5.3.2 Managing Subspaces When Storage Is a Priority

A server program with storage constraints can manage the subspaces by performing all steps to create and delete subspaces each time an application program runs. (See below for an overview of the required steps.) Managing subspaces in this way can reduce storage contention in the system, but is much more costly in terms of server performance.

5.3.3 Creating a Single Subspace

The following is a simple illustration of how a server program can manage a single subspace.

Macro Used	Reason
STORAGE OBTAIN	To obtain storage in the application server address space. Receives the storage to be used for subspaces.
IARSUBSP IDENTIFY	Make storage ranges eligible for subspaces. Specify the storage that was previously obtained.
IARSUBSP CREATE	Create the subspace. Receive the STOKEN.
ALESERV ADD	Add the subspace to the DU-AL, specifying the STOKEN. Receive the ALET.
IARSUBSP ASSIGN	Assign the range of storage that a program running in the subspace can reference. Specify the STOKEN and the storage range.
BSG	Branch to subspace. Specify the ALET.

Run application program in subspace

BSG	Branch back to full address space addressability. Specify ALET 0.
IARSUBSP UNASSIGN	Disassociate the range of storage from the subspace. Specify the STOKEN and the storage range.
ALESERV DELETE	Remove entry from the access list. Specify the ALET.
IARSUBSP DELETE	Delete the subspace. Specify the STOKEN.
IARSUBSP UNIDENTIFY	Make the storage ranges ineligible for subspace usage. Specify storage that was previously specified on IARSUBSP IDENTIFY
STORAGE RELEASE	Release storage in application server address space. Specify storage.

For a more complete example, please refer to Appendix B, “Sample Program for Subspace Management” on page 197. Additional information on the setup and management of subspaces can be found in *MVS/ESA SP V5 Extended Addressability Guide* and *MVS/ESA SP V5 Auth Assembler Services Reference ENF-ITT*.

5.3.4 Exploitation

The subspace group facility has been developed to enhance the capabilities of service provider subsystems such as CICS/ESA.

Since this storage isolation and protection can be implemented through CICS/ESA and not the application, little or no modification should be required to existing CICS application programs and code. Customer investments in

application programs are protected, and requirements for staff and resources to implement the function should be minimal. The benefits of program isolation and storage protection are provided while overall cost of implementation is minimized.

CICS/ESA 4.1 uses the subspace group facility to implement the transaction isolation function. This function extends storage protection, using the subsystem storage protection function introduced in CICS/ESA 3.3. The CICS/ESA transaction isolation function now offers storage protection between application programs, ensuring that one application program does not accidentally overwrite the storage of another application. Transaction isolation preserves data integrity within an address space, reduces system outages, improves systems and change management, and enhances CICS/ESA application development and maintenance.

The use of this facility can result in a reduction of the application development and maintenance staff required to support the subsystems that use subspace groups. Systems that use the subspace group facility should see improved availability since outages that could be caused by inadvertent and accidental access to storage belonging to another application will be eliminated. Improving subsystem and application availability should result in improved end-user productivity.

Chapter 6. Dispatcher Enclaves

New functional enhancements provide subsystems and authorized programs the ability to create groups of SRBs, called enclaves, to give them any dispatching priority and to make them preemptable.

Enclaves allow application servers to more accurately control and report on resources used when satisfying requests for service whose goals differ from overall goals of the server itself.

This support allows work such as CPU-intensive or low-priority routines to execute asynchronously and to be scheduled with a priority independent of the home address space. It also allows SRBs scheduled on behalf of a client to run in a server address space but have CPU service time associated with the client address space.

6.1 Background on Dispatcher Restructure

In previous releases, the dispatcher searched for work on a number of different queues. SRBs were located on global (system related) or local (address space related) SRB dispatching queues. TCBs, asynchronous exits, and lock promotion requests were located by searching individual address spaces. The individual priorities of work units were tied directly to address space priority.

MVS/ESA SP 5.1.0 dispatcher restructure changed the method used to locate and dispatch work. The dispatcher now only searches a system work unit queue or WUQ, and finds all of its work on this queue. A WUQ can have any mix of SSRBs, SRBs, TCBs, asynchronous exit requests, and lock promotion requests. Each one of them is represented by a commonly addressable control block known as a work element block or WEB.

The WEBs on a WUQ represent ready work and are in priority order. When the dispatcher looks for work, it removes a WEB from the head of a WUQ and attempts to run the work. If the work is interrupted, its WEB is placed back on the WUQ but now behind all other WEBs with equal priority. This grants more equitable access to processor resource by automatically rotating units of work of a given priority.

This new dispatching structure provides for more independence and granularity on assigning single work unit priority.

The separation of work unit dispatching from ASCB dispatching allows MVS to relax the tie between work unit priority and ASCB priority.

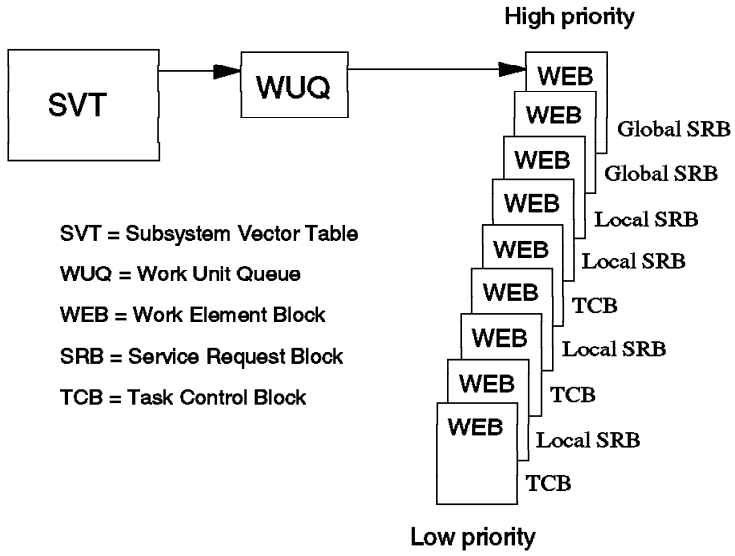


Figure 52. New Dispatching Queue Structure

6.2 SRB Routine Enhancements

MVS/ESA SP 5.2.0 asynchronous scheduling services include the following enhancements:

- New macro interface which simplifies SRB management
- Enhanced control of SRB dispatching priority
- Optional preemption of SRB routine

6.2.1 IEAMSCHD Macro and Service Routine

Prior to MVS/ESA SP 5.2.0, in order to schedule an SRB routine, a program was required to supply the SRB control block to the SCHEDULE macro interface and take care of later storage freemain using a resource manager termination routine (RMTR) or a functional recovery routine (FRR). This approach was error-prone and forced a rigid SRB structure.

The new IEAMSCHD macro and its service routine allow simpler SRB management.

- The caller is no longer required to provide the SRB control block nor routines to free its storage.
- The caller sets the *minor* dispatching priority of the SRB using the PRIORITY parameter.

Programmers are advised to use the IEAMSCHD macro from now on, as it is IBM′s intention to enhance only this macro and its service routine. The

SCHEDULE macro interface will remain only for compatibility with previous releases of MVS.

6.2.2 Preemptable-Class SRB Routines

Prior to this release, once an SRB routine received control on a processor, it was allowed to complete no matter whether higher priority work was waiting. It could be suspended under certain situations, but never preempted by other work.

Using the IEAMSCHD macro service, the caller now has the option of allowing the SRB routine to be preempted by a time slicing algorithm similar to the one used for tasks. Some time later, it will be redispatched at its current priority. In the mean time, other work with higher or equal priority can get access to the processor.

There are three types of preemptable-class SRB routines:

- Client SRB relates to an SRB that runs on behalf of another address space that requested the service.
- Enclave SRB denotes it pertains to an enclave (enclaves are explained later in this chapter).
- Preemptable SRB belongs to the preemptable-class but is neither client nor enclaved.

There are differences on how they are prioritized.

6.2.3 SRB Prioritization

Nonpreemptable-class SRB routines can be scheduled at the standard SRB global or local priority. Preemptable-class SRB routines can be scheduled to run above, below, or intermixed with tasks in the same or other address space.

Dispatching priority of a work unit now contemplates *major* and *minor* dispatching priority.

- All nonpreemptable SRBs and preemptable-class SRBs derive their major priority from their home address space′s priority.
- Client SRBs derive their major priority from their client address space.
- Enclave SRBs derive their major priority from their enclave.

The minor priority of any preemptable-class SRB routine derives from the PRIORITY keyword on the IEAMSCHD macro used to schedule it. It is equivalent to task dispatching priority.

6.3 Enclave Services

Enclave services provide the mechanism to associate service request work units together, and give them a common dispatching priority, independent of the address space that scheduled the SRB. These related SRB routines form groups, or enclaves, whose processor resource consumption is managed collectively by the SRM, according to the dispatching policy defined for that enclave, which is completely independent of the priority of the home address space of the individual SRB routines or the address space that created the enclave.

6.3.1 Creation and Deletion of an Enclave

The enclave macros, IWMECREA and IWMEDELE, provide an authorized user with the ability to create and delete an enclave. An authorized user can use the IEAMSCHD macro to associate SRB routines to that enclave.

A unique token, called an enclave token or ETOKEN, is created for each enclave, usually by the IWMECREA macro, and used on subsequent IEAMSCHD and IWMEDELE macro invocations to identify a particular enclave.

The address space that creates an enclave is the owner of that enclave, and the CPU time used by all of the SRB routines in it will be accumulated in the owner′s address space for SMF purposes. An owner is generally a subsystem or server address space providing some system-wide service, and must have previously connected to WLM via the IWMCONN macro.

When an enclave is deleted or terminated, outstanding SRBs are allowed to complete as normal, but converted to ordinary preemptable SRBs associated with their home address space for all purposes. No further SRBs may be scheduled into the deleted enclave.

6.3.2 Participant Address Spaces

An address space is a *participant* when there are SRBs scheduled into it that are associated with an enclave the address space does not own.

An owner address space of an enclave, has the possibility to schedule SRB routines into its own address space, and also into a participant address space that performs some specialized processing for the subsystem to help satisfy the work request.

The ENV keyword of the IEAMSCHD macro is provided for this facility.

6.3.3 Enclave Prioritization

Enclave major priority is equivalent to address space priority, and will propagate to all SRBs scheduled into it. The system administrator defines a dispatching policy and service classes to be associated with enclaves.

When SRM detects that the enclave is to move to the next performance period, every SRB routine in the enclave will have its major priority changed automatically to reflect the new performance period. Those SRBs that are executing at the time of the performance period switch will be allowed to complete their time slice before changing their priority.

As mentioned previously, SRB routine minor priority is set via the IEAMSCHD macro, and is unaffected by SRM.

6.3.4 PURGEDQ Interactions

The purging of preemptable-class SRB routines is similar to the purging of SRB routines that utilize the suspend with token services. PURGEDQ will ensure that, while PURGEDQ processing is taking place, no new SRB routine matching the purge parameters will begin execution.

If the target address space of the purge is the current primary address space, then PURGEDQ will:

- Wait for all nonpreemptable-class SRBs that match purge parameters to either complete execution or become suspended by suspend with token services
- Wait for all executing preemptable-class SRB routines, which match purge parameters, to either complete execution, become preempted, or become suspended by suspend and token services

Regardless of the target address space of the purge, control is given to the RMTR for those SRB routines that match the purge parameters and have not yet started execution, no matter what their preemptability.

6.4 SRM and WLM Considerations

A new function is provided to associate performance objectives with each enclave. If the system is operating in WLM compatibility mode, the performance objectives are given by a dispatching definition in the IPS and ICS members of SYS1.PARMLIB.

In the ICS, the system programmer is allowed to specify control performance groups associated with SUBSYS and SRVCLASS. If not specified, then enclaves will be put in the same performance group as the owner address space.

There are no new IPS parameters for this support. The parameters on the control performance group definition in the IPS will define the dispatching priority for the SRBs associated with the enclave.

There is no MPL control provided, as swapping is not included in this support. Also no storage isolation controls are provided as enclaves do not own storage.

If the system is operating in WLM goal mode, performance objectives for enclaves are in service class definitions defined via the WLM administrative interface. Several new attributes are added to the IWMCLSFY macro to increase the flexibility to cluster work associated with an enclave, and changes are made to the WLM administrative application to allow these attributes to be used by installation defined subsystem types. Furthermore, a new subsystem type (distributed data facility or DDF) is predefined and enabled for these new attributes.

Even in compatibility mode, the WLM service classes and classification rules must be defined, and the relevant WLM policy must be activated. This is required since the ICS assigns the performance group based on that information.

For performance monitoring purposes, WLM supports two interfaces, IWMRCALL and IWMRQRY, to provide monitors with information about performance related events. IWMRCOLL provides service class (goal mode) or performance group (compatibility mode) related information. IWMRQRY provides address space related information.

6.5 SMF Reporting

SMF provides reporting on the enclave based on the subsystem and service class information that has been set up for the enclave in the IPS and ICS members of SYS1.PARMLIB.

The SMF Type 30 record has been enhanced to report on enclave CPU time. Two new fields are added to the processor accounting section:

- SMF30ASR contains the CPU time accumulated by preemptable and client SRBs for this job.
- SMF30ENC contains the CPU time for enclave SRBs.

In order to reduce the impact to existing accounting packages, the above times are also added to the TCB time in SMF30CPT, as preemptable-class SRBs activity probably replaces what was previously or is otherwise executed under a task.

Four new fields are added to the Performance Section:

- SMF30ETA for enclave transaction active time
- SMF30ESR for enclave session residency time
- SMF30ESU for enclave CPU service units
- SMF30ETC for enclave transaction count

Note that the above may combine work requests from multiple service classes or performance groups, and may differ from the service class or performance group associated with the address space.

Once the ICS is changed to isolate enclave activity, the reports will show a reduction in the subsystem′s address space period consumed and a corresponding increase in whatever periods are associated with enclave work. A similar change occurs in goal mode.

The total wall clock time for enclave activity should be a starting point for defining goal mode objectives.

6.6 Exploitation

Prior to enclave services, SRB routines were not preemptable. Once they received control of the processor, they generally retained control until they finished processing. DB2 queries are typically CPU intensive, and therefore, in DB2′s effort to manage potentially thousands of SRB routines, the processor resource would be easily overutilized. Other work, possibly of higher priority, would then be CPU starved. Additionally, since SRB routines were selected for execution by the priority of their home address space, SRB routines scheduled on behalf of a low priority TSO user would have a priority equal to the DB2 address space. Thus, a TSO user with low priority could submit a complex query and be given more access to system resources than he or she should receive.

By disassociating the SRB routine priority from its home address space and by allowing certain types of SRB routines to be preemptable, client and enclave SRBs allow DB2 to show a significant performance improvement for certain types of complex queries. The distributed data facility allows DB2 queries to come from other systems. DB2 is the home address space for these queries and will be able to manage the priority of these cross-system queries, and dispatch them at a priority that is appropriate for the query, and not DB2′s address space priority.

Coding Sample.

See Appendix C, “Coding Sample for Enclave Implementation” on page 199.

Reference Books

Detailed information on using enclaves and a description of IWMxxxxx macros is found in the *MVS/ESA SP V5 Workload Management Services*. Usage and coding of the IEAMSCHD macro is explained in the *MVS/ESA V5 Authorized Assembler Services* manuals.

Chapter 7. Shared Pages

Shared pages is a new MVS function, provided by the IARVSERV macro services, that permits more than one virtual storage page to simultaneously share the same system resource. This means that at any given time, multiple virtual pages could use the same real frame, the same expanded storage frame, or the same slot on auxiliary DASD. The shared storage can be used to share data and programs among programs, or for communication between programs. Multiple virtual views are possible without requiring multiple physical copies in system resources. MVS/ESA SP 5.2.0 is exploiting this option for various system functions, and it is also available for use by application programs.

The sharing of data can save significant storage in many types of applications. Data is available to all applications with no increase in storage usage. Furthermore, the nature of the shared access could optionally be more restricted for one virtual page than for another. Using the IARVSERV parameter UNIQUEWRITE, modified shared storage is copied on a changed page increment only. With another parameter, SHAREDWRITE, all sharing programs have the updates at the same time and all can modify them.

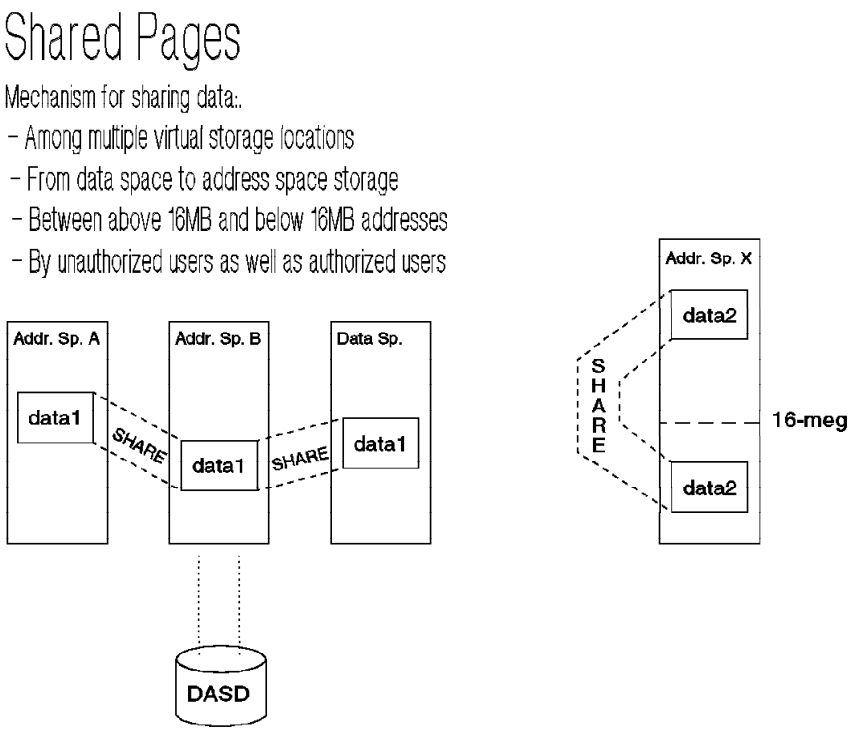


Figure 53. Shared Pages

7.1 Hardware and Software Requirements

Software requirements

- MVS/ESA SP 5.2.0 and above

Hardware performance support

A copy-on-write hardware facility is provided for additional performance improvement when using the IARVSERV UNIQUEWRITE parameter. The copy-on-write attribute is available when suppression-on-protection is present on the machine. Suppression-on-protection is provided on the following machines:

- S/390 9672 processors
- 9021: 711-based models at sec 228215
- 9121: 511-based models with sec c35954
- 9221: 211-based models

7.2 IARVSERV Macro Services

IARVSERV (RSM virtual storage services) macro services provide the interface to set up data sharing among different virtual storage areas.

The following services are provided:

SHARE Requests that a source of data be made accessible through a given virtual storage area (target). There are three types of data sharing; each is called a specific type of view of the data. This view is the way the program accesses the target virtual storage:

- Read only view - where data may not be modified via the view.
- Shared write view - where data can be modified via the view.
- Copy-on-write (unique write) view - where data modifications will not be seen in other views. Any attempt to modify the shared data in the view will result in a unique copy of the affected page being created for that address or data space.

UNSHARE Requests that the specified virtual storage area (target) no longer shares storage. RETAIN is used with UNSHARE to specify the contents of the target address once the it leaves the sharing group:

- RETAIN=YES - keeps the target area for that view.
- RETAIN=NO - clears the target area with binary zeros (makes first reference).

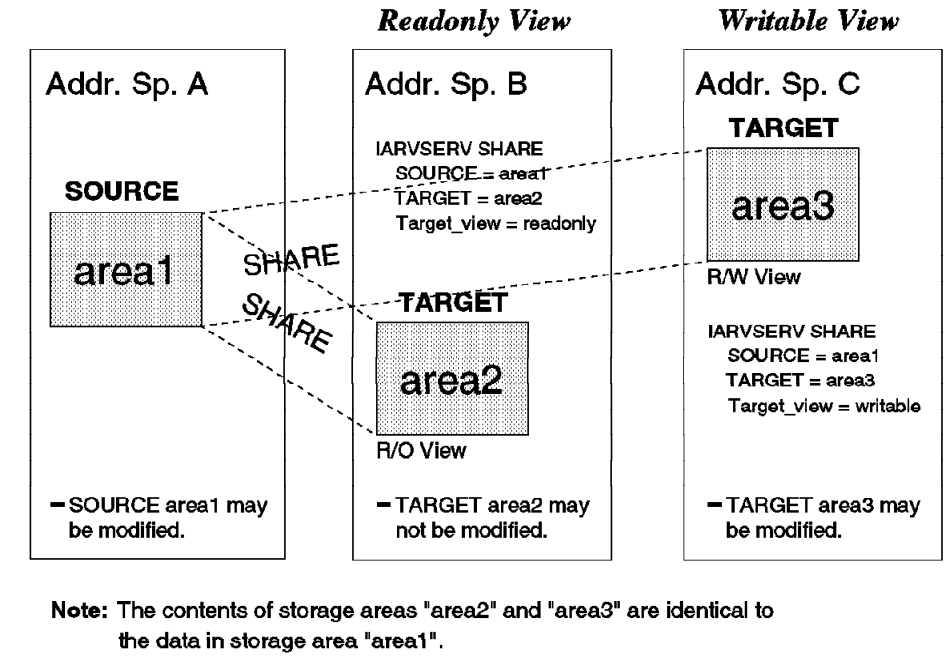


Figure 54. Sharing Views

The data to be shared is called the source. Specifically, this refers to the actual source data in the virtual storage that contains the data. The term target is used to describe the virtual storage area where the source data is made available as shareable.

When IARVSERV SHARE completes successfully, the source and its corresponding target form a sharing group. A sharing group can consist of several target areas, all using the same source data. All sharing of data is actually done on a page (4K) basis. If the source page is already a member of an existing sharing group, the target becomes a member of that existing sharing group. A page is called a sharing page if it is a member of a sharing group.

With IARVSERV macro services, virtual storage can be shared by multiple address spaces or data spaces. Any address that you have valid access to can be used. Hiperspace, a VIO window, a V=R region, or or PSA cannot be used. The target area cannot contain page-protected or page-fixed pages. Storage that is currently allocated as DIV MAP to any DIV object cannot be the target, nor can the source storage have been allocated DIV MAP to a hiperspace.

7.2.1 Address Conversion Using the IARR2V Macro

You may find a time when doing I/O or diagnostic programming that you have the actual central storage address and require the virtual address. To reference data, you need the virtual address.

The macro IARR2V provides the conversion of a central storage address to virtual address. It is used to convert an input real storage address to a virtual storage address, ASID, and STOKEN. The ASID indicates the address space that owns the frame and the STOKEN indicates the address space or data space that uses the frame.

The IARR2V macro replaces existing IARUTRV (IEAPTRV) service. See *MVS/ESA SP V5 Auth Assembler Services Reference ENF-ITT* for details on this macro.

7.2.2 Changkey Service

RSM′s service to change the storage protection key of virtual storage (CHANGKEY macro) is changed to fail if any portion of virtual storage within the storage range is part of a shared data group with read-only or shared-write view.

7.3 RSM Tracing

RSM component tracing is enhanced to include new function trace options. These new functions include options for the new RSM virtual services. The options may be specified when starting RSM component tracing (via the TRACE CT,ON,COMP=SYSRSM system command).

The new RSM component trace options are:

- IARVSERV - Virtual services group. This group is equivalent to specifying VSSHARE and VSUNSHAR.
- VSSHARE - IARVSERV SHARE service.
- VSUNSHAR - IARVSERV UNSHARE service.

7.4 RMF Support for Shared Storage

If an address space chooses to share its storage with other address spaces, the storage can no longer be related to only one address space. RMF offers the following features to report shared pages or shared page groups:

- Monitor I - New measurements in the paging activity report and the workload activity report.
- Monitor II - The shared storage page-in rate related to each address space is included in the page-in rates in the address space state data report (ASD/ASDJ).
- Monitor III - The shared storage page-in rate related to each address space is include in the page-in rates in the following reports:
  - Group response time (GROUP) report
  - Storage delay summary (STORS) report
  - Storage frames (STORF) report
  - Storage variation of the Job report (STORJ)

  The shared page-in delays are included in the common or local page-in delays in the following reports:

- Storage delays (STOR) report
- Storage delay summary (STORS) report
- Detailed storage delays (DSD) user report

There are four new columns in the storage frames (STORF) report that show shared page counts.

7.5 Improved Fork Function

The fork function of MVS/OpenEdition, uses the *unique view shared pages shared data support.*

Fork process prior to MVS/ESA SP 5.2.0

During a fork operation, MVS copies one process, called the parent process, into a new process, called the child process. MVS then places the child process in a new address space, the forked address space, provided by an APPC/MVS transaction initiator. This copying is done by using MVCL operation.

Fork process with MVS/ESA SP 5.2.0

With MVS/ESA SP 5.2.0 *shared pages unique view support,* the costly MVCL operation is no longer needed. RSM will create a copy of the parent page from the shared range and give it to the child. This process makes the fork process much faster.

7.6 UCB Virtual Storage Constraint Relief Exploitation

UCB VSCR support exploits the shared pages by using the ability to access control blocks residing above 16MB using 24 bit pointers. Each device defined requires 24-bit common area virtual storage for a unit control block (UCB). The limit of 4096 device numbers was removed in MVS/ESA SP 5.1.0, but then 24-bit virtual storage constraints was the limiting factor on the number of devices.

The UCB VSCR support provides relief for this virtual storage problem by allowing the UCB to be defined in 31-bit storage above 16MB. Old interfaces using 24-bit UCB pointers are still maintained; this is accomplished through the use of shared pages support. For more information, see the chapter Chapter 8, “UCB Virtual Storage Constraint Relief” on page 139.

Chapter 8. UCB Virtual Storage Constraint Relief

The *UCB VSCR* (Virtual Storage Constraint Relief) support provides relief for below 16MB virtual storage constraint by allowing the UCB to be defined in 31-bit storage above 16MB. It does so in such a way that compatibility with old interfaces using 24-bit UCB pointers are maintained. This is accomplished through the use of shared pages support. The UCB VSCR support:

- Allows 4-digit device support to be fully exploited
- Removes device definition constraints caused by below 16MB common virtual storage consumed by UCBs
- Moves the UCBs into storage above 16MB
- Maintains compatibility with existing interfaces
- Improves systems management of the IODF in a SYSPLEX environment

IOS is changed to build the UCB either above or below 16MB based on the specifications in the I/O Definition File (IODF) and in the UIMs. The UCB is above 16MB if both the UIM allows it and the customer defines it to the IODF by specifying in the HCD I/O device list LOCANY equal to yes. That is, if the device support code does not support UCBs above 16MB, the installation cannot define devices above 16MB.

Control of UCB Residency

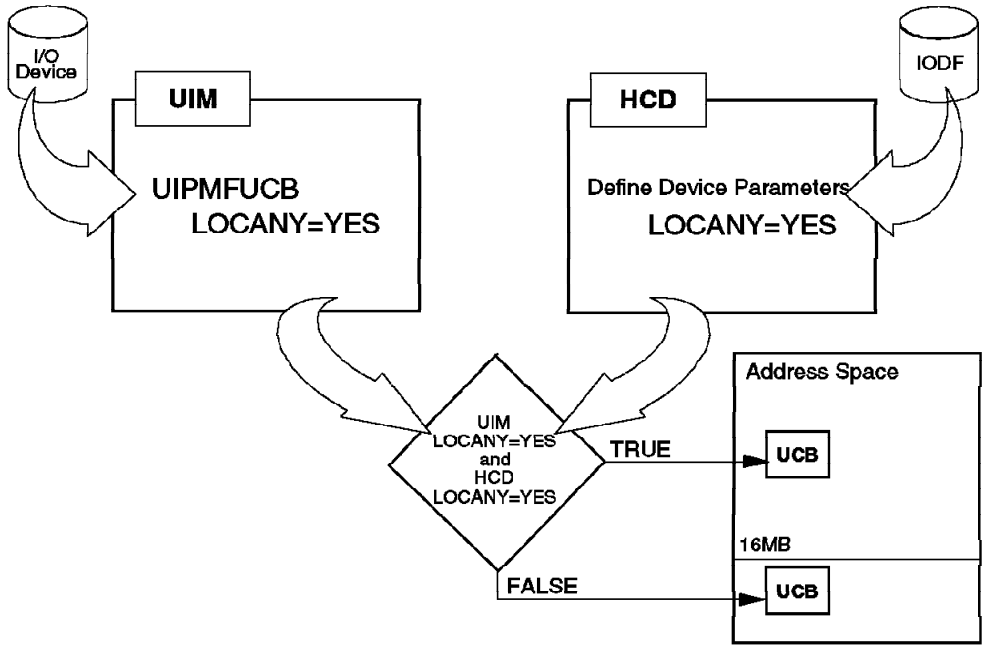


Figure 55. Control of UCB Residency

IOS ensures that UCBs above 16MBs do not cross page boundaries. For application programs that require 24-bit addresses, the system enables above 16MB UCBs to be accessed in below 16MB private, virtual storage. During allocation, the system automatically creates a below 16MB view into the actual above 16MB UCB. The view is known as a captured UCB. It enables an application to access the UCB in the private storage of its address space. The system automatically captures an above 16MB UCB at allocation and releases the UCB at deallocation.

With dynamic allocation, the user can choose not to capture a UCB if affected applications can handle above 16MB UCBs.

This support can also simplify systems management for a sysplex configuration by using the same I/O definition file (IODF) for every system in the sysplex. Collecting all the device definitions in the sysplex in a single IODF is possible because this support increases the number of devices you can define on each system without constraining common storage below 16MB.

8.1 Moving UCBs Above 16MB

In MVS/ESA SP 5.1.0, the UCB prefix was moved into storage above 16MB to provide VSCR and 4-digit support. This move was unconditional.

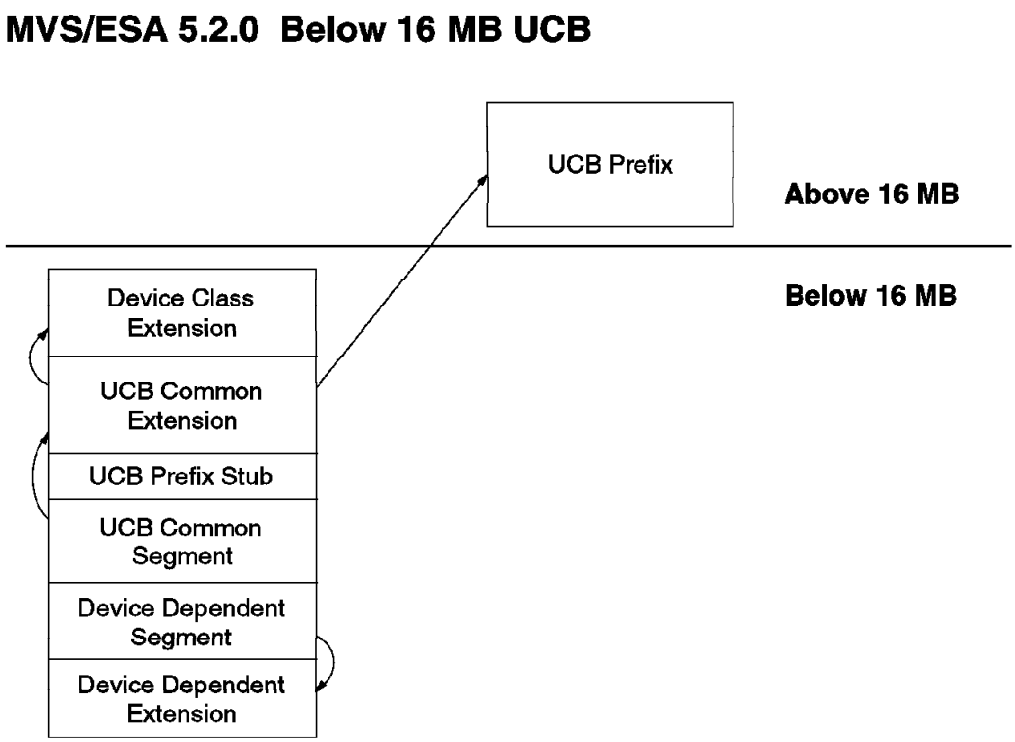


Figure 56. Below 16MB UCB

All programs that accessed fields in the UCB prefix had to use the pointer to the UCB prefix using services provided in MVS/ESA SP 5.1.0 .

Now, with MVS/ESA SP 5.2.0 and DFSMS/MVS 1.3, you are able to move the rest of the UCB above 16MB.

As it was mentioned before, you have to keep in mind that, when you move the UCB to storage above 16MB, programs using the 24-bit UCB pointer will not be able to address it. To maintain compatibility with existing interfaces that have a dependency on a 24-bit UCB pointer, the UCB VSCR support allows capturing of the following UCB segments:

- UCB common segment (UCBOB)
- Device dependent segment
- Device class extension (DCE)
- UCB common extension

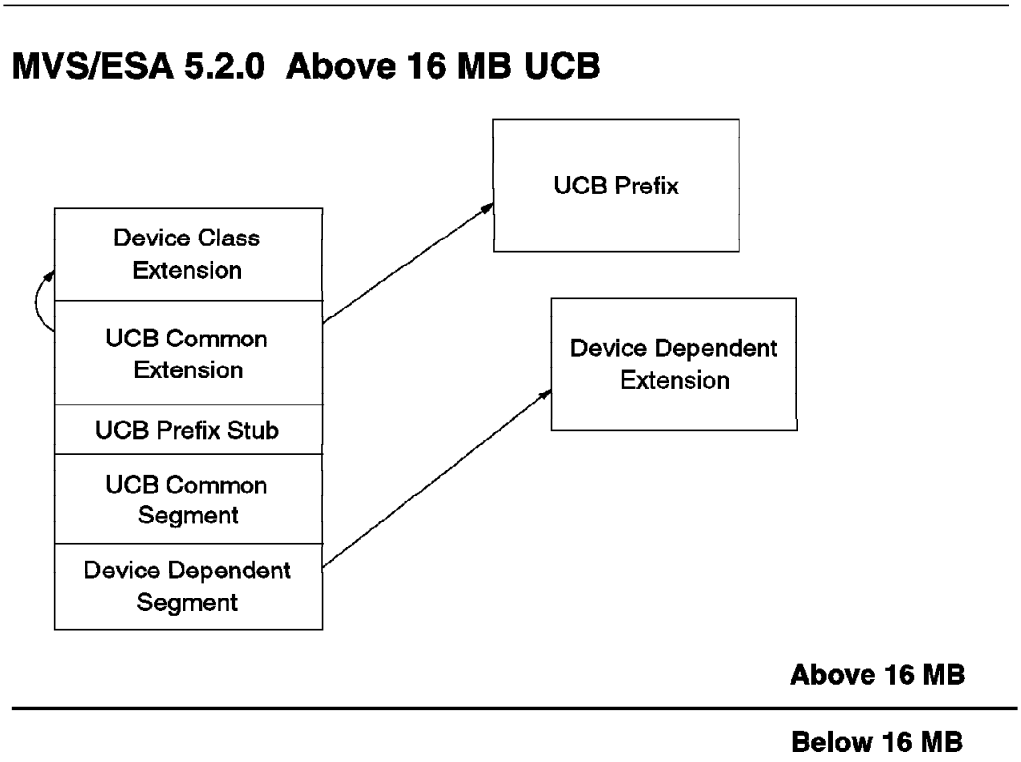


Figure 57. Above 16MB UCB

Some considerations about the UCB extensions when the UCB is placed in storage above 16MB:

- UCB common segment no longer points to the UCB common extension. The pointer to the UCB common extension (UCBEXTP) is only a 24-bit pointer, which cannot be used to reference above 16MB storage. The address of the UCB common extension must, therefore, be obtained using a service.
- The device dependent extension is not contiguous with the rest of the UCB.
- The move of the UCB segments, except for UCB prefix, is conditional at the option of the installation.

The service used to obtain ths address of the UCM common extension is called IOSCAPU. Figure 58 on page 142 shows the UCB for device BF3 below 16MB and Figure 59 on page 142 shows the same UCB after it was moved above 16MB.

```
UCBPRFIX: 00F36E08
    -0008  LOCK..... 00000000  IOQ...... 00000000
UCBOB: 00F36E10
    +0000  JBNR..... 00        FL5...... 88        ID....... FF
    +0003  STAT..... 04        CHAN..... 0BF3      FL1...... 40
    +0007  FLB...... 20        NXUCB.... 00F36E90  WGT...... 06
    +000D  NAME..... BF3       TBYT1.... 30        TBYT2.... 30
    +0012  DVCLS.... 20        UNTYP.... 0F        FLC...... 00
    +0015  EXTP..... F36DE8    VTOC..... 00000000  VOLI..... 00000000
    +0022  STAB..... 00        DMCT..... 00        SQC...... 00
    +0025  FL4...... 00        USER..... 0000
UCBCMXT: 00F36DE8
    +0000  ETI...... 00        STI...... 00        FL6...... 01
    +0003  ATI...... 40        SNSCT.... 20        FLP1..... 22
    +0006  STLI..... 00        FL7...... 00        IEXT..... 01F95320
    +000C  CHPRM.... 00        SATI..... 00        ASID..... 0000
    +0011  WTOID.... 000000    DDT...... 00FCDD84  CLEXT.... 00F36DB8
    +001C  DCTOF.... 0000
UCBXPX: 01F95320
    +0000  RSTEM.... 00        MIHKY.... 04        MIHTI.... 00
    +0003  HOTIO.... 00        IOQF..... 00000000  IOQL..... 00000000
    +000C  SIDA..... 0000      SCHNO.... 0000      PMCW1.... 0018
    +0012  MBI...... 02A0      LPM...... 00        LPUM..... 00
    +0017  PIM...... 00        CHPID.... 00000000  00000000
    +0020  LEVEL.... 01        IOSF1.... 40        IOTKY.... 00
```

Figure 58. UCB for Device BF3 Below 16MB

```
UCBPRFIX: 01C4D2D0
    -0008  LOCK..... 00000000  IOQ...... 00000000
UCBOB: 01C4D2D8
    +0000  JBNR..... 00        FL5...... 88        ID....... FF
    +0003  STAT..... 04        CHAN..... 0BF3      FL1...... 00
    +0007  FLB...... 20        NXUCB.... 00000000  WGT...... 06
    +000D  NAME..... BF3       TBYT1.... 30        TBYT2.... 30
    +0012  DVCLS.... 20        UNTYP.... 0F        FLC...... 00
    +0015  EXTP..... C4D2B1    VTOC..... 00000000  VOLI..... 00000000
    +0022  STAB..... 00        DMCT..... 00        SQC...... 00
    +0025  FL4...... 00        USER..... 0000
UCBCMXT: 01C4D2B0
    +0000  ETI...... 00        STI...... 00        FL6...... 01
    +0003  ATI...... 40        SNSCT.... 20        FLP1..... 22
    +0006  STLI..... 00        FL7...... 00        IEXT..... 01E66F90
    +000C  CHPRM.... 00        SATI..... 00        ASID..... 0000
    +0011  WTOID.... 000000    DDT...... 00FCDD84  CLEXT.... 01C4D280
    +001C  DCTOF.... 0000
UCBXPX: 01E66F90
    +0000  RSTEM.... 00        MIHKY.... 04        MIHTI.... 00
    +0003  HOTIO.... 00        IOQF..... 00000000  IOQL..... 00000000
    +000C  SIDA..... 0000      SCHNO.... 0000      PMCW1.... 0018
    +0012  MBI...... 02A0      LPM...... 00        LPUM..... 00
    +0017  PIM...... 00        CHPID.... 00000000  00000000
    +0020  LEVEL.... 01        IOSF1.... 40        IOTKY.... 00
    +0023  MIHFG.... 00        LVMSK.... 00000001
  Actual UCB Common segment address 01C4D2D8
  Device is dynamic
```

Figure 59. UCB for Device BF3 Above 16MB

The installation can dynamically add or modify devices to reside either above or below 16MB. This allows the user to dynamically change the residency of a UCB without re-IPLing the system; the device must be offline and defined as dynamic in the IODF. UCBs above 16MB may be static, installation static, or dynamic. They may have either 3-digit or 4-digit device numbers.

The following table shows the possible residency of various pieces of the UCB.

Table 5. UCB Segments and Affect of the UCB VSCR

UCB Segment	Size	Current location	Move Above 16MB	Capture	Copies supplied
UCB Prefix (minus stub)	40 bytes	Above 16MB	N/A	No	Yes
UCB Common (with Prefix stub)	40 bytes	Below 16MB	Yes	Yes	Yes
Device Dependent Segment	Device Dependent	Below 16MB	Yes	Yes	Yes
UCB Common Extension	32 bytes	Below 16MB	Yes	Yes	Yes
Device Dependent Extension	Device Dependent	Below 16MB	Yes	No	No
Device Class Extension	Device Dependent	Above or Below 16MB	Yes	Yes	Yes
Device Characteristics Table		Below 16MB	No	N/A	No
Device Statistics Table	10-bytes per entry rounded to 8-byte multiple	Above or Below 16MB	No	N/A	No
Device Descriptor Table		Below 16MB	No	N/A	No

The table shows the various UCB related control blocks, their sizes (where applicable), their current residency, whether UCB VSCR will allow the residency changed to above 16MB (where applicable), whether those moved above 16MB can be captured (where applicable), and whether copies of them are returned through UCBSCAN.

UCBs above 16MB are not accessible from the chain of UCBs chained off the CVT. The UCB services introduced with dynamic I/O reconfiguration must be used to locate a UCB above 16MB (UCBLOOK or UCBSCAN). Even if the UCB is captured, it does not appear on the CVT chain of UCBs. In addition, captured UCB addresses are not returned by the execution of the UCBLOOK or UCBSCAN macros services.

Devices that contain SYS1.LINKLIB, SYS1.SVCLIB, or SYS1.LOGREC are captured into common storage rather than private storage. Each of these three datasets has a DCB pointed to out of the CVT, which may be used by programs running in any address space. If the DEBs pointed to out of those DCBs contained private rather than common addresses for the UCB, all such programs would not execute properly. Therefore, all such UCBs must be captured to common storage. The installation must be aware of this characteristic when moving such devices above 16MB. In essence, the pages in which these UCBs reside use virtual storage in every address space.

To reduce the impact, do one or both of the following when you define the configuration with HCD:

- Define the SYS1.LINKLIB, SYS1.SVCLIB, and logrec data sets on the fewest number of devices possible to reduce the number of UCBs that the system captures in common storage
- Define the devices containing these data sets in sequential device number order in HCD to minimize the number of pages of storage the system captures. Such sequential numbering can increase systems management effort if the installation also needs to maintain availability and performance goals by having a given group of devices span multiple control units.

8.2 Hardware Supported

All devices that are supported by MVS/ESA SP 5.2.0 are also supported by UCB VSCR. It allows all IBM-supported devices to have their UCBs defined above 16MB as device support allows (UIM modules). Potentially, a few non-strategic IBM devices might require their UCBs to be below 16MB.

8.3 Software Requirements

Software requirements

- MVS/ESA SP 5.2.0 is the first release that supports UCB VSCR.
- DFSMS 1.3 is the first release with the required UIM modules to support UCB VSCR.
- Device Support Facility (ICKDSF) R16 (5655-257) with APAR PN66540.

Program product for full function:

- To define a JES-managed device as an above 16MB UCB, you must install JES2 SP 5.2 or JES3 SP5.2.1. For a list of devices currently supporting this function, see *MVS/ESA SP V5 HCD: Planning*.

If installed, level required:

- ACF/VTAM V4 R1 for MVS/ESA (5695-117) with APAR OW08227
- ACF/VTAM V4 R2 for MVS/ESA (5695-117) with APAR OW08227
- ACF/VTAM V3 R4.1 (5685-085) with APAR OW08227
- ACF/VTAM V3 R4.2 (5685-085) with APAR OW08227
- ACF/TCAM V2 R4 (5735-RC3) with APAR OW10464 to prevent TCAM′s use of DASD devices that are defined with above 16MB UCBs.
- ACF/TCAM V3 R1 (5665-314) with APAR OW10465 to prevent TCAM′s use of DASD devices that are defined with above 16MB UCBs.

Related program products:

- To place the RACF data base on a device with an above 16MB UCB, you must install one of the following:
  - RACF V1 R9 with PTF UW90115
  - RACF V1 R9.2 with PTFs UW90115 and UW90116
  - RACF V2 R1 with PTF UW90114
  - RACF V2 R2

Exploiting program products:

- MVS/DITTO V2 R1 (5665-370) with PTF UN67088

- DITTO/ESA V1 R1 (5695-100)
- ESCM V1 R3 (5688-008) with APAR PN67103
- GAM V1 R3.1 with APAR PN69441
- DATABASE 2 (DB2) V2 R3 (5665-DB2) with a PTF
- DB2 V3 R1 (5685-DB2) with a PTF
- DB2 V4 (5695-DB2)

8.4 Benefits of Captured UCBs

The determination of the need for UCB VSCR is either the need to define additional devices in the system, as for a sysplex, or the need for more space in the below 16MB private area. The UCB VSCR support allows you to grow your I/O configuration beyond the limits of virtual storage constraints. This allows the I/O configuration to keep up with both processor growth and the requirement for symmetric configurations in a parallel sysplex environment. So it allows the definition of up to the architectural limit of 64K devices and provides the ability to increase the amount of storage below 16MB available to the system.

If a sufficient number of UCBs are moved above 16MB to allow the private area to expand, you generally get a benefit from the UCB VSCR support. The private area can be consumed by UCB VSCR as UCBs are captured, but the 1MB increase in the size of the private area provides a large buffer for these captured UCBs before they would begin to degrade any below 16MB private area storage constraints. Figure 60 shows the storage map with UCBs below 16MB, and Figure 61 shows UCBs above 16MB.

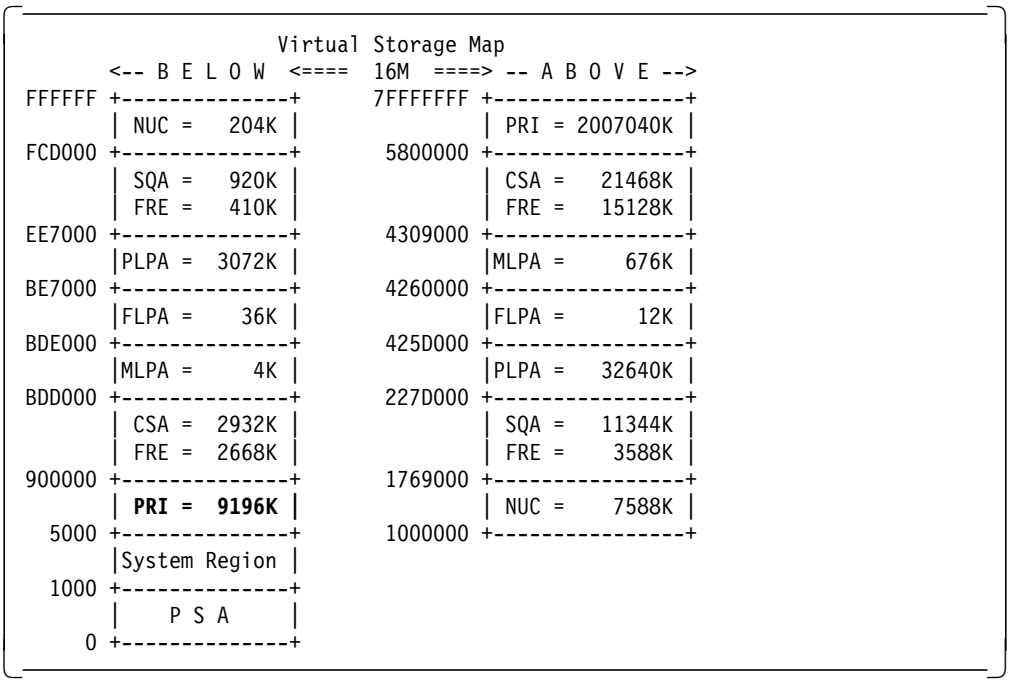


Figure 60. Virtual Storage Map with UCBs Below 16MB

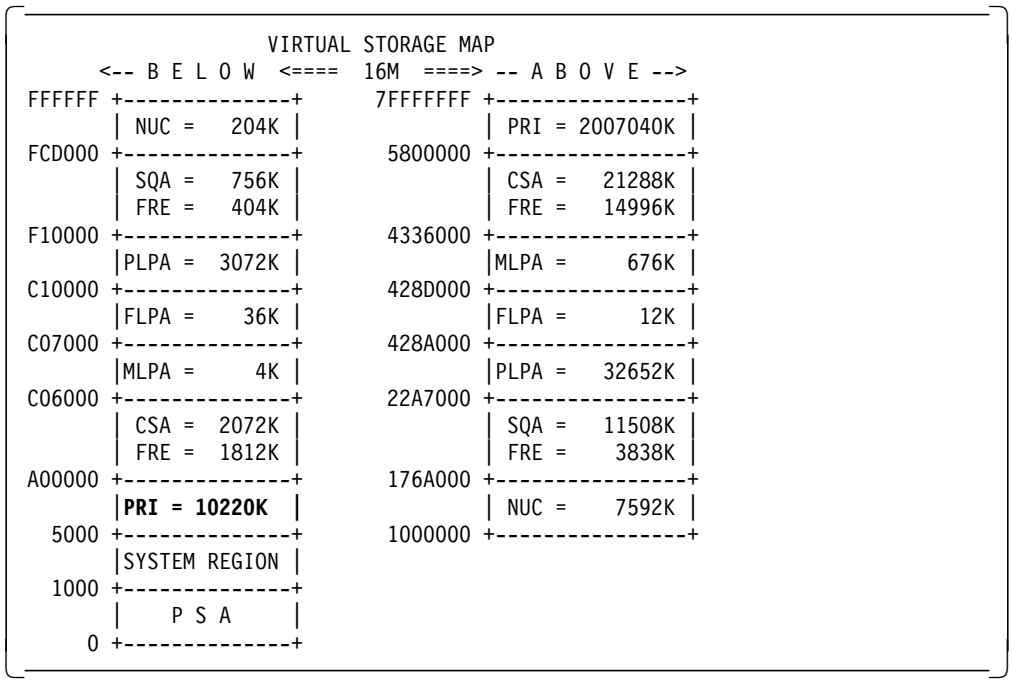


Figure 61. Virtual Storage Map with UCBs Above 16MB

If the address space is common storage constrained, your installation gets a benefit so long as there is sufficient private storage available to accommodate any captured UCBs created in that address space. Captured UCBs are created and destroyed during device allocation and deallocation and reside in below 16MB private storage for the duration of a device allocation. The UCB VSCR allows UCBs above 16MB to be used transparently with existing applications. The captured UCB is created by exploiting the RSM shared pages support. It allows multiple virtual addresses to access the same real storage. These virtual addresses can either be above or below 16MB.

The UCBs that are moved above 16MB can therefore be accessed through a 24-bit window when they cannot be accessed directly. The net result is multiple virtual addresses resolving to the same real address. The location of these 24-bit windows is important. If they were located in common virtual storage below 16MB, the benefits of moving the UCB above 16MB would be lost. The UCB would be captured back into the storage where it would have resided anyway if it had never been moved above 16MB. Rather than using below 16MB common storage to capture the UCBs above 16MB, below 16MB private LSQA storage is used. This effectively allows the use of captured UCBs by one address space to be isolated from all others. Any given address space is free to capture only those UCBs it requires without causing storage constraints in other address spaces.

IOS provides the mechanism to create and destroy captured UCBs. This UCB capture mechanism is provided through a new programming interface IOSCAPU. IOS manages all captured UCBs on behalf of an address space. No UCB is captured more than once in any given address space and the details of this processing is transparent to the caller.

8.5 UCB VSCR Programming Dependencies

The changes required of programs to support UCB VSCR are dependent upon the location from which UCB pointers are obtained, UCB fields accessed, and actions taken with the UCB address. Programs with all of the following characteristics support UCB VSCR with no changes:

- Allocate devices via JCL or dynamic allocation (SVC 99)
- Obtain the UCB addresses through allocation interfaces (TIOT) OPEN interface (DEB), or from UCBLOOK/UCBSCAN when pinning/unpinning is used
- Obtain UCB address from CVTSYSAD and are AMODE 31
- Obtain UCB address from ECVTNUCP and are AMODE 31
- Do not reference fields in the UCB common extension
- Do not pass UCB addresses between address spaces

Programs that do not allocate devices “normally ”or access fields in the UCB common extension need to change to support UCB VSCR. Programs obtaining the UCB pointer from UCB services may need to change depending upon the desired access - actual UCB or captured UCB address. Programs that obtain UCB addresses from places other than allocation interfaces, the OPEN interface, or UCB services such as the DCQ, CVTUCBAD, and UCBNXUCB do not support above 16MB or captured UCBs without changes.

If you define above 16MB UCBs, evaluate and test each application for its dependency on below 16MB UCBs. For example, you might find the following dependencies:

- Programs that access the device class extension (DCE) in the UCB and do not support 31-bit mode must change to access the DCE with the IOSDCXR service. Programs that can access the actual above 16MB DCE directly, can continue to use the DCE pointer, UCBCLEXT, in the UCB common extension.
- Programs that reference the UCB common extension or UCB prefix extension must use the following supported services. The parameters and services provided with MVS/ESA SP 5.2.0 are indicated.

  UCB Common Extension:

  – Use the COPY option of the UCBSCAN macro with the CMXTAREA parameter to obtain a copy of the UCB common extension. Any caller can issue this macro, including callers in problem or supervisor state and callers in 24-bit or 31-bit addressing mode.

  – Use the UCBLOOK or (starting with MVS/ESA SP 5.2.0) UCBSCAN ADDRESS macro with the UCBCXPTR parameter to obtain the address of the segment. The address returned by UCBLOOK or UCBSCAN ADDRESS is always the actual, not captured, address.

  – When performance is important, use the IOSCMXA or (starting with MVS/ESA SP 5.2.0) IOSCMXR macro to obtain the address of the segment. The caller must provide recovery. IOSCMXA and IOSCMXR return a captured, common extension address for an input captured address and return an actual, common extension address for an input actual address.

In all these cases, use the UCBCMEXT DSECT in the IEFUCBOB macro to map the UCB common extension.

UCB Prefix Extension:

– Use one of the following to obtain a copy of the UCB prefix extension:

  - The COPY option of the UCBSCAN macro with the UCBPAREA parameter
  - The PRFXDATA option of the UCBINFO macro
  - The UCBPAREA option of the UCBLOOK macro

  Any caller can issue UCBSCAN COPY or UCBINFO, including callers in problem and supervisor state and callers in 24-bit or 31-bit addressing mode.

– Use the UCBLOOK or (starting with MVS/ESA SP 5.2.0) UCBSCAN ADDRESS macro with the UCBPXPTR parameter to obtain the address of the UCB prefix extension.

– When performance is important, use the IOSUPFA or (starting with MVS/ESA SP 5.2.0) IOSUPFR macro to obtain the address of the segment. The caller must provide recovery.

When you obtain a copy of a UCB prefix extension, use the IOSDUPI mapping macro to map the prefix extension. When you obtain the address of a UCB prefix extension, use the IOSDUPFX mapping macro to map the prefix extension. See *MVS/ESA SP V5 Auth Assembler Services Guide* for a comparison of these macros.

• For programs that pass UCB addresses between address spaces, you must either:

  – Ensure that the programs can handle actual above 16MB addresses

  – Change the programs to pass device numbers instead

  Actually, such programs could also pass a common, captured UCB. However the other options are recommended, because they do not require common storage below 16MBs.

  – Programs obtaining the UCB address from UCBLOOK or UCBSCAN ADDRESS must change to use the LOC=ANY parameter to receive above 16MB UCBs. Programs using UCBSCAN COPY do not need to change to obtain copies of above 16MB UCBs because the copies are returned in storage accessible by the caller.

  – Programs calling UCBDEVN and passing an above 16MB UCB must run in AMODE 31-bit. If the caller runs in AMODE 31-bit and passes a 24-bit UCB pointer, the pointer must have a clean high order byte.

  – Programs running in AMODE 24 and calling the DFSMS/MVS MSGDISP macro with the TEST=YES parameter cannot pass an actual above 16MB UCB. These programs must pass either an actual below 16MB UCB or a captured UCB.

  – Programs calling EDTINFO must change to use the LOC=ANY parameter to receive device numbers for above 16MB UCBs.

  – To reserve a device with a 31-bit UCB address, programs calling RESERVE with the UCB parameter must change to also use the LOC=ANY parameter or pass a captured UCB address.

- To release a device with a 31-bit UCB address programs calling DEQ with the UCB parameter must change to also use the LOC=ANY parameter or pass a captured UCB address.
- To access above 16MB UCBs directly, programs performing dynamic allocation must change to set the new S99ACUCB bit on in the S99FLAG1 field when building the SVC 99 parameter list. Use this capability if applications can access above 16MB UCBs directly for example, with VSAM. Note that applications using other access methods require 24-bit UCBs and cannot access above 16MB UCBs directly.
- When dynamic allocation specifies that a UCB not be captured, programs retrieving UCB addresses from the task input/output table (TIOT) must change to obtain UCB addresses using the IEFDDSRV macro.
- Programs not using normal allocation interfaces but still requiring 24-bit UCB addresses must explicitly capture and release UCBs using the IOSCAPU macro.
- Programs using the IOSINFO macro must pass a below 16MB actual or captured UCB. For an above 16MB actual UCB, programs can use the IOSUPFA or IOSUPFR macro to obtain the UCB prefix extension for the subchannel number of the UCB.
- With a future release of DFSMS/MVS installed, the fields for tape devices in the device-dependent extension move to the UCB device-dependent segment (DDS) and the device-dependent extension pointer is zero. If a tape device has an above 16MB UCB, stop using the device-dependent extension pointer, reassemble the program, and access the fields in the DDS. If a tape device has a below 16MB UCB, you can continue to use the DDE pointer or access the fields directly in the DDS.

Other application programs that use intended interfaces are not affected. Unintended interfaces have changed; you should evaluate and test each application for its dependency on below 16MB UCBs.

- If you define above 16MB UCBs and have programs that listen for ENF signals 23, 24, 25, 27, 28, 29, or 30, be aware that these programs can now receive above 16MB UCB addresses.

8.5.1 UCB VSCR Differences for Tape UCBs

Changes to above 16MB tape UCBs:

- The DDE has been merged with the DDS for tape UCBs above 16MB.
- The UCBXTNB address is zero in above 16MB tape UCBs.

The structure of above 16MB tape UCBs is different than that of below 16MB tape UCBs. In the case of above 16MB tape UCBs, the DDE has been merged with the device dependent segment (DDS). However, the layout in storage of both the below and above 16MB tape UCBs is the same. The same new mapping macro can therefore be used to access both versions.

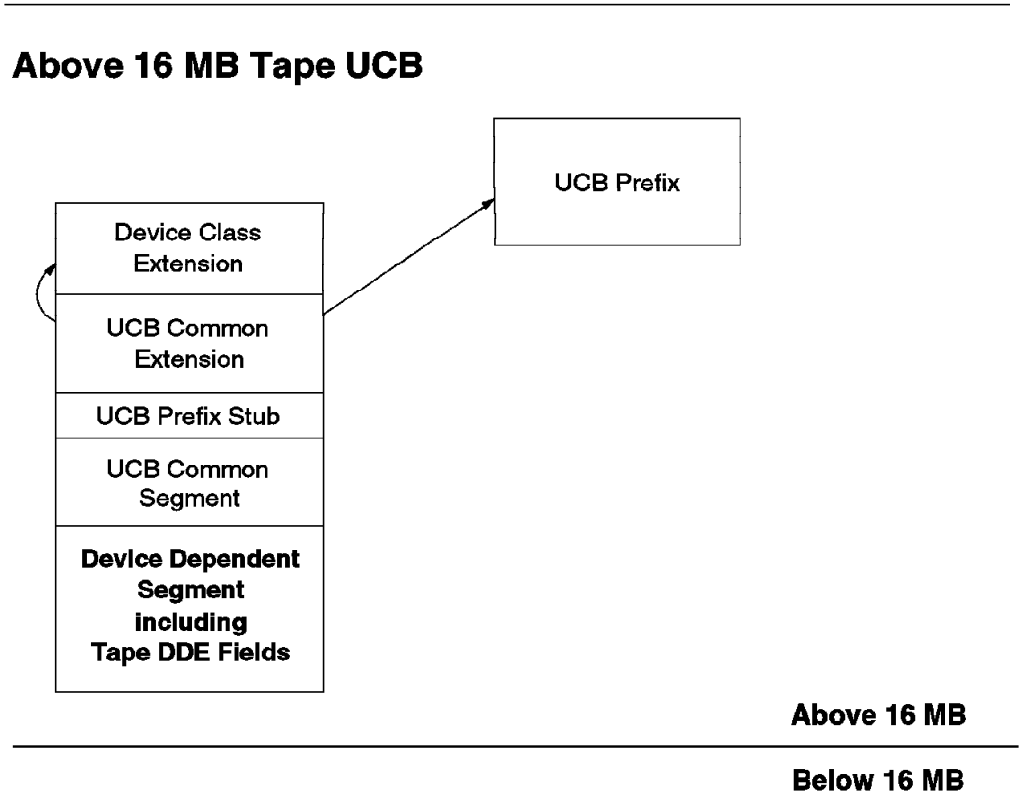


Figure 62. Above 16MB Tape UCB

The pointer to the Tape DDE (UCBXTNB) is a three-byte pointer out of the DDS. The tape DDE could not be accessed for above 16MB UCBs in this case. The solution chosen allows the tape DDE to be referenced as an offset from the UCB common segment (like DDS) by using a new mapping for IECDUCBT. The UCBXTNB address is zero in above 16MB UCBs. THE UCBXTNB address is still valid for below 16MB UCBs allowing either the old or new mapping to be used in this case.

8.6 UCB VSCR Compatibility with Existing Programms

A typical program takes the following steps when accessing a device:

1. **Allocation:**
   The DD statements in the step that executes the program define the resources that the program needs to access. Allocation associates these logical resources with physical devices and volumes. Allocation places the UCB address in a 3-byte field, TIOEFSRT. The high order byte of that word contains status flags. The TIOT is one of the allocation control blocks constituting allocation's user interface. The TIOEFSRT is documented as one of the ways programs can obtain the UCB address. Moving the UCB above 16MB without a captured UCB would require the TIOEFSRT to expand to a 4-byte field, which would force changes to all programs accessing that field. Increasing the TIOT entry size would also reduce the number of DDs supported. With the captured UCB design, the above 16MB UCB is captured

when the device is allocated. This allows the UCB to be accessed via a 24-bit address. The 24-bit captured UCB address is placed in the TIOEFSRT field. Thus, no change to programs accessing the TIOT is required.

2. **HLL Program I/O:**
   When a high level language (HLL) program issues an I/O, it does not directly reference the UCB and is unaffected by the movement of the UCB above 16MB. The HLL program uses I/O routines built into the HLL.
3. **HLL I/O Routine Issues OPEN:**
   The HLL I/O routine, on behalf of the HLL program, issues the OPEN macro. OPEN obtains the UCB address from the TIOT and places it in a 3-byte DEB field called DEBUCBA. The high-order byte of this word contains the device modifier file mask. Like the TIOTEFSRT field, the DEBUCBA field is a documented way for programs to get the UCB address. Similarly, moving the UCB above 16MB would cause all programs accessing this field to change. The 3-byte captured UCB address is placed in the DEBUCBA field and the DEB remains unchanged.
4. **HLL I/O Routine Issues Read or Write:**
   The HLL I/O routine on behalf of the HLL program invokes one of the DFP access methods, such as QSAM and BSAM. The DFP access methods, with the exception of VSAM and Media Manager, use the DEB and are executed in 24-bit mode (AMODE 24). Moving the UCB above 16MB, without captured UCBs, would require major changes to the access methods.
5. **Access Method Issues EXCP:**
   The DFP access method issues the EXCP macro on behalf of the HLL I/O routine. EXCP also uses the DEB and runs AMODE(24). Moving the UCB above 16MB, with no captured UCBs, would also require major changes to EXCP.
6. **EXCP Issues STARTIO:**
   EXCP then issues the STARTIO macro to pass the I/O request to IOS. IOS can handle 4-byte UCB addresses and runs AMODE(31). It can, therefore, directly reference the above 16MB UCB. The captured UCB address is converted to the actual above 16MB UCB address and the I/O request started.
7. **I/O Interrupt Processing:**
   When the I/O interrupt is presented, IOS receives the address of the actual UCB. If that UCB was captured, IOS ensures that the DIE is entered in the address space specified in IOSASID and that it is passed the captured address. The driver must set IOSASID to that ASID where the STARTIO was issued.
8. **Post Status:**
   Post status is then scheduled to the address space specified in IOSASID. Post status invokes the driver-end exits. If the UCB is captured, the driver end-exits receive the captured UCB address. The driver must set IOSASID to that ASID where the STARTIO was issued.
9. **Close:**
   The DEB is deleted by CLOSE processing.
10. **Unallocation:**
    The device is no longer associated with the program. The UCB address is removed from the TIOT, and if the UCB was captured, the captured UCB is deleted.

The above example covered cases where programs obtain the UCB address from the TIOT or the DEB. There are several other ways programs can obtain UCB addresses. Programs can access the CVT fields CVTUCBA to access the chain of UCBs or CVTDCQA to access the device class queue, which contains pointers to the first UCB in each device class. In addition, the IOSVSUCB and the IOSLOOK services can be used to find UCB addresses.

Note: The CVT, IOSVSUCB, and IOSLOOK methods of looking up UCB addresses were replaced in MVS/ESA SP 4.1.0 with the UCB services UCBLOOK and UCBSCAN macro services. These new services are the only way to find both dynamic and 4-digit UCBs. They are also the only way to find above 16MB UCBs.

8.7 UCB VSCR Implementation

HCD allows devices to be defined above 16MB only if supported by the device support code. For each IBM supported device whose device support code allows above 16MB UCBs, the corresponding UIM is changed to indicate that the UCB can reside above 16MB. UIMs for devices not supporting above 16MB UCBs do not require changes. If you request the UCB for a device to be defined above 16MB, but the UIM does not allow it, the UIM overrides the request, and the UCB is defined below 16MB.

The HCD default is for all UCBs defined in a pre-MVS/ESA SP 5.2.0 level IODF to reside below 16MB. Therefore, if an installation does not need VSCR for UCBs and wishes to keep UCBs below 16MB, they simply have to specify UCBs below 16MB for new devices defined in a MVS/ESA SP 5.2.0 level IODF.

Installations can define UCBs above 16MB in HCD with a “group change” command. However, the installation needs to be sensitive to:

1. Any devices for which you have written your own UIMs; if you want the UCBs for these devices to be above 16MB, you need to change your UIMs to specify UCBs above 16MB.

   Note: The UIM for the DUMMY device type allows UCBs above 16MB. You can specify these UCBs above 16MB in HCD.

2. Any devices used by your applications or vendor programs that require actual UCBs to reside below 16MB. For programs that use captured UCBs, the actual UCBs can be above 16MB and do not fall into this category.

   You need to specify in your HCD definition that the UCBs for these devices are to be defined below 16MB.

Since your installation may need to modify some applications to support UCB VSCR, staged changes to the I/O definition may be required. Initially, your I/O definition should have all the UCBs below 16MB; then UCBs can be migrated above 16MB. With dynamic I/O definition, these changes can be made without a re-IPL. However, if a large portion of the I/O configuration is affected by a dynamic change, a correspondingly large portion of the workload on the system might have to be quiesced to allow the dynamic change to proceed. Allocated and online devices cannot be changed dynamically.

You may use the following sequence to move UCBs above 16MB:

1. Move defined but not attached devices dynamically.
2. Move noncritical devices dynamically.
3. Move critical devices, like Catalog, Sysres, and other system volumes.

Try to move most of the devices dynamically to avoid a re-IPL.

Note: Be aware of the differences between above 16MB tape UCB and below 16MB tape UCB.

8.8 HCD - UCB VSCR Implementation

HCD 5.2.0 is a new HCD release shipping as a component of MVS/ESA SP 5.2.0. It is delivered on the JES2 and JES3 MVS SP product tapes as separately installable FMIDs. HCD 5.2.0 runs on any processor supported by MVS/ESA SP 5.2.0. All devices supported by MVS/ESA SP 5.2.0 can be configured by HCD 5.2.0.

One of the new functions that is supported by HCD 5.2.0 is UCB VSCR. This allows customers to define UCBs to reside in 31-bit addressable storage, above 16MB. Whether or not the UCB resides in the 31-bit storage is controlled by the the LOCANY parameter.

```
  Goto  Show/Hide _Filter  Backup  Query  Help
------------------------------------------------------------------------------
                              I/O Device List       Row 1 of 2071 More:  1 of 2
Command ===> filter_____________________________________ Scroll ===> PAGE

Select one or more devices, then press Enter. To add, use F11.

Configuration ID . . : LO6RMVS1     Sysplex systems

  -------Device-------      Loc  -------Control Unit Numbers + ---------
/ Number Type +        Dyn. any  1--- 2--- 3--- 4--- 5--- 6--- 7--- 8---
_ 001A   9032          Y    _    001A ____ ____ ____ ____ ____ ____ ____******
_ 001B   9032          Y    _    001B ____ ____ ____ ____ ____ ____ ____
_ 001C   9032          Y    _    001C ____ ____ ____ ____ ____ ____ ____
_ 001D   9032          Y    _    001D ____ ____ ____ ____ ____ ____ ____
_ 0030   2741C         _    _    0005 ____ ____ ____ ____ ____ ____ ____
_ 0031   2741C         _    _    0005 ____ ____ ____ ____ ____ ____ ____
_ 0032   2741C         _    _    0005 ____ ____ ____ ____ ____ ____ ____
_ 0033   2741C         _    _    0005 ____ ____ ____ ____ ____ ____ ____
_ 0034   2741C         _    _    0005 ____ ____ ____ ____ ____ ____ ____
_ 0035   2741C         _    _    0005 ____ ____ ____ ____ ____ ____ ____
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F5=Reset     F7=Backward
 F8=Forward   F9=Swap      F10=Actions  F11=Add      F12=Cancel   F13=Instruct
F20=Right    F22=Command
```

Figure 63. HCD Frame with the New LOCANY Parameter

Device parameters and features can be changed via the *Change* or *OS Group Change* function that is available on all I/O device lists. For *Change*, only the device definition can be selected. For *OS Group Change*, only devices belonging to the same device group (DASD, TERMINAL) can be selected.

```
------------------------------------------------------------------------
           Change Device Group / Operating System Configuration
------------------------------------------------------------------------
               Define Device Group Parameters / Features        Row 1
 Command ===> _____________________________________________ Scroll ===> PA

 Specify or revise the values below.

 Configuration ID . : LO6RMVS1     Sysplex systems

 Parameter/  Value    P Req.  Description
 Feature
 OFFLINE     No               Device considered online or offline at IP
 DYNAMIC     ________         Device supports dynamic configuration
 LOCANY      Yes              UCB can reside in 31 bit storage
 ALTCTRL     No               Separate physical control unit path
 SHARED      Yes              Device shared with other systems
 SHAREDUP    No               Shared when system physically partitioned
 ****************************** Bottom of data ************************




  F1=Help        F2=Split       F4=Prompt      F5=Reset       F7=Backwa
  F8=Forward     F9=Swap       F12=Cancel
------------------------------------------------------------------------
```

Figure 64. OS Group Change Function

To provide the same support for the LOCANY parameter as for the DYNAMIC parameter in previous releases, the *attribute group change* has been extended to allow updating the LOCANY parameter for all selected devices at once. Figure 65 shows an example.

```
  Goto  Show/Hide  Filter  Backup  Query  Help
- ---------------------- Actions on selected devices --------------------
  |                        Attribute Group Change                       |
C |                                                                     |
  | For all devices in the selected group, choose whether ...           |
S |                                                                     |
  | 3   1.  Allow dynamic configuration ........... DYNAMIC=YES        |
C |     2.  Do not allow dynamic configuration .... DYNAMIC=NO         |
  |     3.  UCB can reside in 31 bit storage ...... LOCANY=YES         |
  |     4.  UCB can not reside in 31 bit storage .. LOCANY=NO          |
/ |                                                                     |
( |                                                                     |
_ |                                                                     |
_ |  F1=Help    F2=Split   F9=Swap   F12=Cancel                         |
_ ----------------------------------------------------------------------
_ 0104   3390          _    _    0100 0101 ____ ____ ____ ____ ____ ____
_ 0105   3390          _    _    0100 0101 ____ ____ ____ ____ ____ ____
_ 0106   3390          _    _    0100 0101 ____ ____ ____ ____ ____ ____
) 0107   3390          _    _    0100 0101 ____ ____ ____ ____ ____ ____
_ 0108   3390          _    _    0100 0101 ____ ____ ____ ____ ____ ____
_ 0109   3390          _    _    0100 0101 ____ ____ ____ ____ ____ ____
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F5=Reset     F7=Backward
 F8=Forward   F9=Swap     F10=Actions  F11=Add      F12=Cancel   F13=Instruct
F20=Right    F22=Command
```

Figure 65. HCD Attribute Group Change

Chapter 9. APPC/MVS Enhancements

APPC/MVS is an implementation of the Advanced Program-to-Program Communication (APPC) protocols in the MVS/ESA operating system based on the LU 6.2 Architecture (see Figure 66).

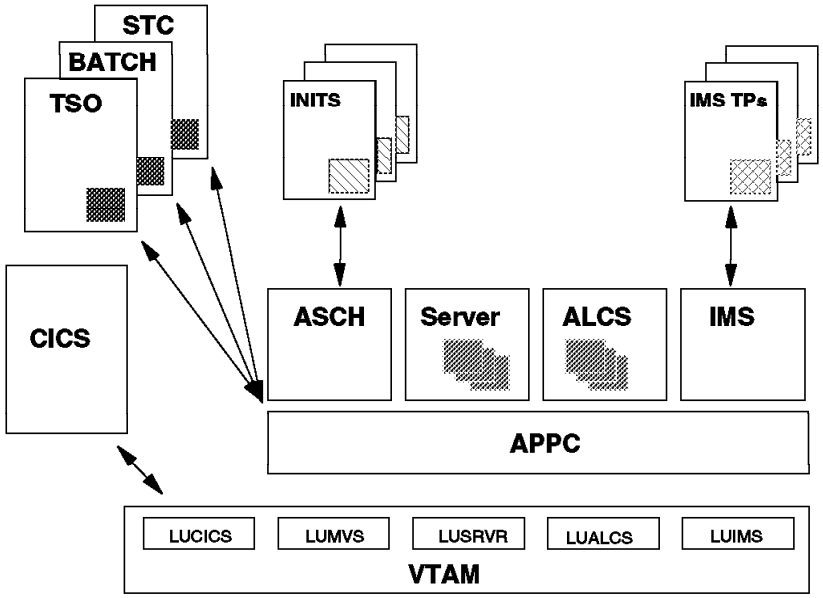


Figure 66. APPC Implementation with APPC/MVS

APPC/MVS was first launched with MVS/ESA Version 4.2 and has been enhanced continuously since then. Applications running in batch as a started task (STC) or in a TSO/E address space can use APPC through callable services provided by APPC/MVS. Applications can use verbs either conforming to the *Common Programming Interface for Communications* (CPI-C) or conforming to a so called *native* interface provided by APPC/MVS. Programs using CPI-C have the highest degree of portability among systems conforming to the SAA standards. It should be mentioned that CPI-C is adopted by X/OPEN for their portability guide XPG4. So using CPI-C does not necessarily limit the portability to SAA systems.

These verbs are implemented as callable services and can be accessed from various programming languages like COBOL, PL/1, REXX, and C. Even assembler can be used to write a transaction program (TP). The implementation is such that a programmer who implements a distributed application based on APPC needs only very few SNA or VTAM skills. All network oriented tasks are done by the implementation under the covers on behalf of the calling program (see Figure 67 on page 156).

As an example, you can think of an application running in a TSO/E address space that communicates to an application running on a PS/2. APPC/MVS

provides this application with the means to communicate with the partner program located on the PS/2 system.

APPC/MVS not only provides for *outbound* conversations as seen from MVS, but also provides for a way to start applications on behalf of a remote application initiation. So it is possible to run an application on a PS/2 that communicates to a partner program running under control of APPC/MVS.

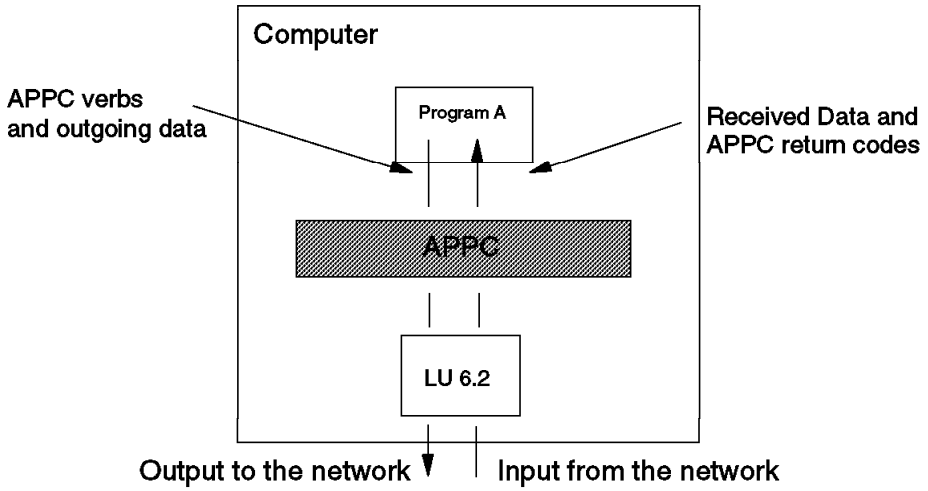


Figure 67. APPC Data and Command Flow

MVS/ESA SP 5.1.0 brings further enhancements that are directed either to the systems programmer or the application developer. They allow for better use of resources assigned to and used by APPC/MVS and easier implementation of applications.

9.1 Diagnosis Enhancements

APPC as an implementation of the LU 6.2 Architecture hides most of the communication services needed to run a distributed application. Due to this hiding, most of the very complex error situations are gathered in a few return codes. The CPI-C interface delivers return codes only, whereas the native interfaces can provide additional reason codes that give more detailed information to the application programmer. This may be either in the form of numbers again or as detailed messages. The latter is the case with APPC/MVS in MVS/ESA SP 5.1.0. This function, implemented with the callable service *ATBEES3* Error_Extract service, which is a synonym for ATBEES3 in this book, helps to diagnose problems in TPs more quickly and easily than one can do with return_codes only. Error_Extract provides reason codes and messages that describe errors that the *local* system (APPC/MVS and VTAM) finds, and error log information and a product set ID for errors that a remote TP or system finds and reports. Error_Extract returns error information only for the last APPC TP

conversation service or CPI-C call that completed processing for that conversation.

The format of the Error_Extract is shown in Figure 68. Every parameter shown in the list must be coded.

```
CAll ATBEES3 ( Conversation_id ,
               Service_name Service_reason_code,
               Message_text_length Message_text,
               Error_log_product_set_ID_length,
               Error_log_product_set_ID,
               Error_log_information_length,
               Error_log_information,
               Reason_code Return_code
             );
```

Figure 68. Error_Extract (ATBEES3) Parameter List in PL/1 Style

Parameter	Description
Conversation_id	This is an identification that uniquely identifies the conversation to the transaction program and APPC. It is only known locally. It should be pointed out that for every conversation request an ID is assigned by APPC. So, even in the case where the conversation could not be established, the transaction program can refer to this ID.
Service_name	The service that failed.
Service_reason_code	This is the simple reason code the service already returned to the caller.
Message_text_length	The length of a message from the local implementation of APPC if any is available.
Message_text	The detailed message text.
Error_log_product_set_ID_length	Length for the following field.
Error_log_product_set_ID	To find a detailed description of the contents of these fields, look up MVS Common Subvectors in *SNA Formats* for a description of: • Product Set ID (X′10′) • Product Identifier (X′11′)
Error_log_information_length	The length of the following field.
Error_log_information	Contains a message that describes an error found by a *partner* system or TP. APPC/MVS itself can send error_log_information to partner systems. An alternate transaction scheduler can send error_log_information by providing this information to cleanup_TP (ATBCTP3).

Reason_code Contains additional information about the result of the call to Error_Extract, when the Return_code contains a value other than 0 or 64 (decimal)

Return_code Specifies the result of the Error_Extract.

9.1.1 ATBEES3 Usage Examples

For an example of using this new service, see E.2, “Sample Code to Call ATBEES3” on page 209. This example gives you an understanding of how to use ATBEES3. The code shown in E.1, “Example Used to Call for CPI Communications” on page 209 shows how to link to an error analysis routine based on ATBEES3. Consider the following:

- There is currently no access to ATBEES3 through the REXX ADDRESS APPCMVS function.
- A call to ATBEES3 is only possible directly after the call to the APPC service routine that failed.

In the first example using ATBEES3, we tried to initialize a conversation with a wrong side information entry. The error indication returned by APPC/MVS is *CM_PROGRAM_PARAMETER_CHECK* **1**. Note that this return code is returned by every implementation of the LU 6.2 Architecture. It depends on the implementation whether it provides further information to the application programmer that he can use to analyze the error. Now, using the ATBEES3, we get additional information that clearly indicates which service caused the error **2**. Even more, ATBEES3 provides a message **3** that clearly indicates the cause of our problem: we used an undefined symbolic destination name.

```
 >> CPI-C/ATB command ?
lc
 There are no valid conversations at the moment
 >> CPI-C/ATB command ?

int hhh
 CMINIT failed with return_code CM_PROGRAM_PARAMETER_CHECK      1
 >> CPI-C/ATB command ?

lc
 Conversion  1 800000000000003E failed earlier..
 If you want to know details answer Yes
y
 Extracting Error information for Conversion ID 800000000000003E
 Service which caused caused error CMINIT Reason Code 000000002C:  2

 ATB80044I Value specified on Sym_dest_name parameter is not defined     3
 in active side information data set.
```

Figure 69. ATBEES3 Example 1. Information Returned by ATBEES3 in Case the Symbolic Destination is not Defined in the Active SI

The second example shows the case where a symbolic destination name points to a partner LU that cannot be contacted by VTAM.

```
+ATBATBOUTP1  ** STARTING **
+ATBOUTP1 - ISSUE ALLOCATE CALL
+ATBOUTP1 ALLOCATE RC IS              1   1
+Bad Return Code from ATBALLC
+Will call ATBEES3
+ATBEES3 returned ...
+Service in ERROR ATBALLC
+Service return_code           100
+ATB80100I From VTAM macro APPCCMD: Primary error return code: 0008,  2
secondary error return code: 0000, sense code: 087D0001.
```

Figure 70. ATBEES3 Example 2. Information Returned by ATBEES3 if the Remote LU Cannot be Contacted by VTAM

Again the simple number **1** with which the programmer normally has to deal is translated into a message **2** which provides much more detailed information about the cause of the failure of the ATBALLOC request.

9.2 Conversation Level Pacing

The following terms have to be clearly separated when talking about APPC:

Conversation Two applications that are communicating with each other by means of APPC are said to have a *conversation.* An application program having a conversation using APPC is said to be a *transaction program (TP)* . Both are *partner programs.* Up to some degree, a conversation can be understood as a telephone call between two people.

Session Every conversation runs on a SNA session. A session is the connection between the two logical units on which the applications run. There is a one-to-one relationship between a session and the conversation flowing thru that session. Only one conversation is running on a session at a time. Multiple conversations require multiple sessions between the LUs on which the applications are running.

It should be understood that a conversation links the applications, and a session links the LUs that are involved.

Either the CPI-C interface or the native interface provides the application with a *send* function, which passes data to APPC: in our case to APPC/MVS. Assuming the application did not set SEND_TYPE=SEND_AND_FLUSH, the buffer passed in the CMSEND call is kept in internal buffer controlled by APPC/MVS. This means data sent by the application is not necessarily passed to VTAM or an equivalent communication manager directly. Instead the data is buffered under control of APPC.

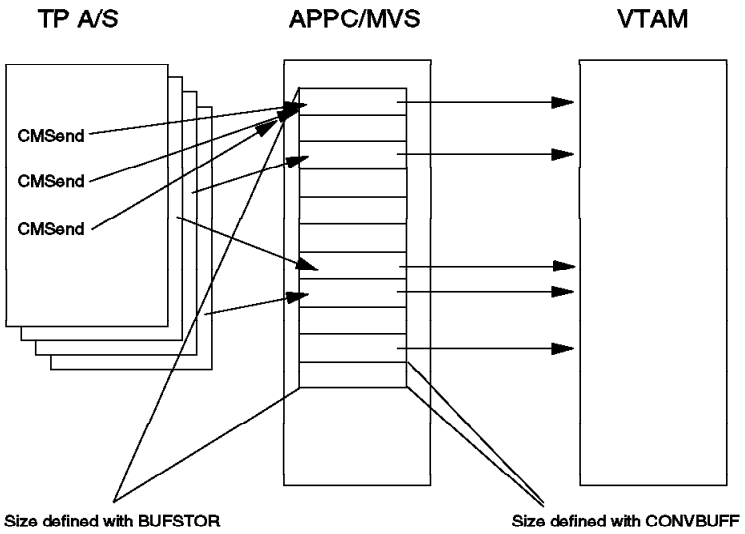


Figure 71. Dataflow in an APPC Environment

When a TP sends data to a partner on an MVS system, the data is stored in buffers until the partner TP receives it. When TPs send large amounts of data and their partners are slow to receive it, more and more buffers of data fill virtual storage. The situation can cause a shortage of buffer space for other TPs and for the system itself.

APPC/MVS provides two means to control the amount of buffer space that TPs use in the APPC/MVS address space (see Figure 71).

BUFSTOR Controls the amount of buffer storage in the APPC/MVS address space that is available to *all* conversations. It prevents using up all of the auxiliary storage when the APPC/MVS address space is paged. BUFSTOR may be an additional start parameter for APPC/MVS:

```
//APPC PROC APPC=00,BUFSTOR=88
//APPC EXEC PGM=ATBINITM,PARM='APPC=&APPC,BUFSTOR=&BUFSTOR'
```

The value is defined in megabytes. If BUFSTOR is omitted from the parameter list, its amount is calculated to be one-third of the available auxiliary storage. It is useful to define BUFSTOR as a parameter on the EXEC statement in the procedure definition of APPC. BUFSTOR may then be easily overwritten at startup time. The value of BUFSTOR may vary from 0 to 2048. The minimum value is 8MB. The value given is rounded down to the next multiple of 8MB if necessary.

CONVBUFF Limits the amount of buffer space available which *any one conversation* can use at a particular time. This parameter should be used when one or more conversations are using so much buffer space that they prevent other conversations from obtaining required buffer space. CONVBUFF prevents one or more

conversations from dominating the system. CONVBUFF can be specified as a further parameter to APPC/MVS:

```
//APPC PROC APPC=00,CONVBUFF=48
//     EXEC PGM=ATBINITM,PARM='APPC=&APPC,CONVBUF=&CONVBUFF'
```

The value given is expressed as kilobytes.

When specifying the amount of buffer space available to any one conversation on the CONVBUFF parameter, keep the following in mind:

- The maximum value you can specify is 2097152KB (decimal).
- If you do not specify a value or specify a value of 0, APPC/MVS uses the default value of 1000 kilobytes.
- If you specify a value between 1 and 39, the system uses a value of 40 (because APPC/MVS requires a minimum of 40 kilobytes of storage per conversation).
- If you specify a value that is greater than the buffer storage limit (which is either the default value of 88 megabyte or the value specified on the BUFSTOR parameter), APPC/MVS uses the BUFSTOR limit.

A new message specifies the buffer size available to a single conversation as shown in Figure 72. The size is rounded up to a multiple of 4K.

```
 COMMAND INPUT ===> /S APPC,APPC=C2,SUB=MSTR,CONVBUFF=56       SCROLL ===> PAGE
0210  S APPC,APPC=C2,SUB=MSTR,CONVBUFF=56
0210  IEF196I PARM='APPC=C2,BUFSTOR=24,CONVBUFF=56'
0210  IEF196I IEF695I START APPC     WITH JOBNAME APPC     IS ASSIGNED TO
0210  IEF196I USER APPCSTC , GROUP SYS1
0210  IEF695I START APPC     WITH JOBNAME APPC     IS ASSIGNED TO USER APPCSTC
       , GROUP SYS1
0210  ATB001I APPC IS INITIALIZING.
0210  ATB014I THE BUFFER STORAGE LIMIT HAS BEEN SET TO 24 MEGABYTES.
0210  ATB016I THE AMOUNT OF BUFFER STORAGE AVAILABLE TO ONE CONVERSATION IS 56
       KILOBYTES.
```

Figure 72. Conversation Buffer Size Definition

9.3 Further Enhancements

There are further enhancements to APPC/MVS that are described briefly in the following sections.

9.3.1 Conversation Correlator Enhancements

A conversation correlator uniquely identifies a conversation from end-to-end as seen from the TP. A TP gets the conversation correlator when accepting the conversation thru Get_Conversation. The initializing client side adds this information to the FMH-5 (function management header) before sending it with the first request unit to the server side.

Note: When APPC/MVS is the partner system, the conversation correlator is always zero. APPC/MVS always provides a conversation correlator that has the value of zero.

With this release of MVS, it is now possible for an user-written scheduler to extract the conversation correlator when it receives an Allocate TP Request message in the IXCMSGI buffer, which is presented to the scheduler by APPC/MVS. The user-written transaction scheduler can use the conversation correlator in case the current conversation has to be protected against outages.

9.3.2 Using Common Dataspaces

MVS/ESA provides applications with the ability to store and share up to 2GB of data in *dataspaces*. Dataspaces can be shared between programs at different extents. The scope of access may be limited to programs running in the same address space as the creator (SCOPE=SINGLE), selected programs in other address spaces (SCOPE=ALL) and to every program running in the system (SCOPE=COMMON). APPC/MVS allows send and receive data buffers to reside in dataspaces and accommodates large amounts of data. The following LU 6.2 conversation services use data buffers and accept an *access list entry token (ALET)* along with the buffer address to designate the address space or data space in which the buffer resides:

- Extract_Information
- Receive_and_Wait
- Receive_Immediate
- Send_Data

The above functions can be accessed in a similar manner either thru the CPI-C set or the native implementation (ATBxxxx) provided by APPC/MVS. Buffers residing in dataspaces are only accessible thru the ATBxxxx set of calls.

Dataspaces are accessed in a controlled manner thru the ALET. For a TP running on MVS 5.1 under control of APPC/MVS, the ALET can represent an entry on the primary access list (PASN-AL) when it points to a SCOPE=COMMON dataspace (also known as a common area dataspace). This kind of a dataspace provides the TP with virtual storage that is addressable from all address spaces and all programs in a system. A TP might want to use a common area dataspace when large amounts of data must be shared across multiple address spaces.

Common area dataspaces are allocated before a TP accesses them. Therefore, whenever an address space starts, its PASN-AL contains an ALET identifying every dataspace that has SCOPE=COMMON. A TP does not need to issue the ALESERV ADD macro to gain access to these dataspaces.

Chapter 10. OpenEdition MVS

Open systems provide their users with a set of functions that underlay common standards. These standards are of several flavors:

- De jure - like POSIX 1003.1
- De facto - like TCP/IP or SNA
- Agreed upon - like DCE

Besides these flavors, standards apply to different building blocks like:

- Application interfaces
- Network interfaces
- Operating system interfaces

In other words, there is no *single* standard that defines *the* open system. In addition there may be a relationship between different standards. For example, Distributed Computing Environment (DCE) uses threads as they are defined in POSIX 1003.1c. So, many standards cannot be implemented without integration of other standards.

The set of functions that must be provided by an operating system is defined in the suite of definitions named 1003.x, or more simply POSIX (Portable Operating System Interface). The first step toward implementing these definitions was done with MVS/ESA 4.3.

Though we are talking about Release 2 of OpenEdition, it is not a product like RMF or TSO/E. OpenEdition covers *all* products that have to be adapted to provide functions that underlay open standards definition. As an example, one can take the hierarchical file system: rather than emulating the hierarchical structure of the files and the byte oriented access method, SMS provides this new filesystem and the necessary access method. All enhancements provided to IBM′s standard products running on MVS/ESA comprise *OpenEdition*.

10.1 OpenEdition MVS

OpenEdition MVS consists of four components.

Table 6. OpenEdition MVS Component List

Component	**FMID**
OpenEdition MVS system services	HOM1120 JOM12N0
OpenEdition MVS application services	HOT1120
OpenEdition MVS shell & utilities	HSU1120 JSU12N0
OpenEdition MVS dbx debugger	HDX1120

This chapter describes the installation process and basic customization of OpenEdition MVS. Full customization and exploitation of OpenEdition MVS is described in detail in the OpenEdition MVS publication library.

10.1.1 OpenEdition MVS System Requirements

Figure 73 shows the minimum list of products that are required to install OpenEdition MVS System Services 1.2.0, install other OpenEdition MVS components, install subsequent maintenance, and operate the OpenEdition MVS environment.

It is strongly recommended that these products be installed and be at a high maintenance level *prior* to installing OpenEdition MVS.

- ACF/VTAM Version 3 Release 4.1 for MVS/ESA (5685-085)
- AD/Cycle Language Environment Version 1 Release 3 (5688-198) or installation of the OpenEdition AD/Cycle C/370 Language support feature
- DFSMS/MVS Version 1 Release 2 (5695-DF1)
- ISPF/PDF Version 3 Release 5 (5665-402)
- RACF Version 2 Release 1 (5695-039)
- SMP/E Version 1 Release 8 (5668-949)
- AD/Cycle C/370 Version 1 Release 2 (5688-216) (Optional)
- DFSMS/MVS Network File System feature (5695-DF1, feature codes 5323 or 5324) (Optional)
- BookManager (R) READ/MVS (5695-046) (Optional)
- RMF Version 5 Release 1 (5655-084) (Optional)
- TCP/IP MVS Version 2 Release 2 (5735-HAL) (Optional)

Figure 73. List of OpenEdition MVS Supporting Products

Note: Installations that use alternative products should consult the appropriate vendor for specific information regarding OpenEdition MVS.

10.1.2 Product Information

All program specific installation information is contained in the Program Directory for MVS/ESA SP 5.1, shipped with your copy of MVS/ESA SP V5.1.

As with all new product installations, it is recommended that you contact the IBM Support Center to obtain the latest preventative service information prior to commencing the installation process. To obtain this information, specify the following *upgrade* and *subset* information.

```
UPGRADE    SUBSET

MVSSP510   HOM1120
MVSSP510   HOT1120
MVSSP510   HSU1120
MVSSP510   HDX1120
```

10.1.3 DASD Storage Requirements

OpenEdition MVS shares data sets with many other IBM products. If you are installing all components of OpenEdition MVS, F.1, “OpenEdition MVS DASD Installation Requirements” on page 211 shows the DASD space required for the OpenEdition MVS target data sets.

10.1.4 Installation Process

As previously mentioned, OpenEdition MVS is dependant on other products. It is recommended that OpenEdition MVS be installed into the same SMP/E zones as these other products, including any of the products marked “optional.”

Note: OpenEdition MVS requires that the name of your AD/Cycle C/370 and AD/Cycle LE data sets be:

- EDC.V1R2M0.xxxxxxxx
- CEE.V1R3M0.xxxxxxxx

If you have used other names, you have to inform OpenEdition MVS of these new names, as shown in 10.5.3, “The Default Profile” on page 190.

The four components of OpenEdition MVS cannot all be installed at the same time. OpenEdition MVS system services must be the first *FMIDs* to be installed as the other components require OpenEdition MVS to be active.

10.1.5 SMP/E and OpenEdition MVS

OpenEdition MVS is installed and maintained using SMP/E, which must be at either:

- V1.8.0 with APAR IR25553
- V1.8.1

OpenEdition MVS introduces a new SMP/E element, ++HFS. This new modification control statement (MCS) is used to define all binary and text data elements that make up the shell and utilities and the dbx debugger features.

During the APPLY process, SMP/E uses the OpenEdition MVS copy utility (BPXCOPY) to install the ++HFS element into a “target library” in the HFS.

During ACCEPT processing, SMP/E installs the ++HFS element as a member of a PDS or PDSE data set. Distribution libraries cannot be part of the HFS.

If you have SMP/E at other release levels or without the correct PTFs, you need to able to receive the FMIDs.

The MVS/ESA SP V5.1 program directory contains detailed descriptions of all the SMP/E steps to install OpenEdition MVS.

10.1.6 Customization

Once OpenEdition MVS system services has been installed (applied), the OpenEdition MVS environment must be prepared before OpenEdition MVS system services can be initialized. This customization involves other products and applications, including RACF V2.1 and APPC/MVS.

10.1.7 APPC Requirements

OpenEdition MVS uses address spaces provided by APPC/MVS. If you have not previously used APPC/MVS, you need to refer to the APPC/MVS documentation *MVS/ESA SP V5 Planning: APPC Management* to assist you in this area.

The specific items that must be customized for OpenEdition MVS are:

1. Create an APPC/MVS transaction program (TP) profile for forked address spaces.
2. Add an LUADD statement for OpenEdition MVS to the APPCPMxx parmlib member.
3. Add a VTAM application definition with the name designated in the LUADD statement.
4. Add a CLASSADD statement for OpenEdition MVS to the ASCHPMxx parmlib member.
5. Start APPC/MVS, including its scheduler address space ASCH

10.1.8 Allocating the OpenEdition MVS Hierarchical File System

OpenEdition MVS requires at least one hierarchical file system (HFS), known as the “root” file system, before it can be started. This HFS data set can be allocated by either JCL, the TSO/E allocate command or ISPF V4.1. DFSMS/MVS Version 1.2 is required to allocate a HFS data set, and the target volume *must* be SMS managed.

If you are using ISPF V4.1 and would like to use option 3.2, you need APAR OW06061.

JCL that can be used to allocate an HFS data set is shown below. The space requirements for the OpenEdition MVS root file system are dependant on your installation′s individual requirements. However, since other components of OpenEdition MVS are installed directly into the root file system, it is recommended that you use approximately 45 cylinders. This saves you having to allocate a new root file system should any maintenance be required.

Even though it is not used by the HFS, you *must* include a nonzero value in the directory space parameter.

```
//JOBNAME  JOBCARD INFORMATION
//*
//STEP01  EXEC PGM=IEFBR14
//HFS       DD DSN=OMVS.ROOT.HFS,SPACE=(CYL,(45,1,1)),
//             DSNTYPE=HFS,
//             DISP=(NEW,CATLG,DELETE),
//             STORCLAS=SMSCLASS
/*
```

10.1.9 Security Requirements

This section assumes that your operating system uses the Resource Access Control Facility (RACF).

The security environment on UNIX systems differs from that typically used on MVS in several ways. If you are unfamiliar with some of the terms used here, a full description of OpenEdition MVS security requirements and their use can be found in *MVS/ESA Planning OpenEdition MVS SP V5*.

Note: The terms *userid* and *uid* have to be carefully differentiated: the first defines a *MVS* user ID whereas the latter defines the identification used within the OpenEdition MVS environment.

The required steps to define an OpenEdition MVS security environment are:

1. Define one or more security administrators as OpenEdition MVS users and assign them OpenEdition MVS *superuser* authority.
   - ALTUSER secadmn OMVS(UID(0) HOME('/') PROGRAM('bin/sh'))
   - ALTGRP SYS1 OMVS(GID(0))
2. Define the OpenEdition MVS kernel by adding the OMVS cataloged procedure to the RACF started procedures table (ICHRIN03) or the STARTED class.

```
DC      CL8'OMVS'       procedure name
DC      CL8'OMVSKERN    userid (can be any userid)
DC      CL8'OMVSGRP'    default group name
DC      XL1'40'         attribute bit
DC      XL7'00'         reserved
```

3. Define the user and group specified for OMVS in the previous step.
   - ADDGROUP OMVSGRP OMVS(GID(1))
   - ADDUSER OMVSKERN DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
4. Define a FACILITY class profile to permit users to query or modify the MVS security environment of a process. These users are known as *daemons*. Enter the command shown:
   - RDEFINE FACILITY BPX.DAEMON UACC(NONE)

   Note:

   You may have to enter the SETROPTS command to activate this facility class.
5. Make the user ID assigned to the kernel a daemon user ID.
   - PERMIT BPX.DAEMON CLASS(FACILITY) ID(OMVSKERN) ACCESS(READ)
6. Add user ID BPXROOT as a superuser.
   - ADDUSER BPXROOT DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
7. Create a uid for the installing user as a superuser
   - ADDUSER SYSPROG DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
8. Create a generic RACF profile to cover all of the intended HFS dataset names and permit update access to the OMVSKERN user ID.

If you are using the new RACF STARTED class to define started procedures, instead of ICHRIN03, you need to use the following commands.

- RDEFINE STARTED OMVS.* STDATA(USER(OMVSKERN) TRUSTED(YES))
- CONNECT OMVSKERN AUTHORITY(CREATE) GROUP(STCGRP) UACC(NONE)
- SETROPTS RACLIST(STARTED)
- SETROPTS RACLIST(STARTED) REFRESH

10.1.10 Starting OpenEdition MVS

OpenEdition MVS runs as a started procedure and uses installation defined parameters for initialization. It is likely that you will want to customize some of the OpenEdition MVS parameters for your environment. The OpenEdition MVS parameters are found in the SYS1.PARMLIB member *BPXPRMxx* and are described in *MVS/ESA Planning OpenEdition MVS SP V5*.

During the SMP/E APPLY process, a sample OpenEdition MVS started procedure was installed into SYS1.SAMPLIB as member OMVS. This may require little or no customization.

```
//OMVS   PROC   OMVS=00
//OMVS   EXEC   PGM=BPXINIT,PARM='&OMVS'
//IEFPARM DD    DSN=SYS1.PARMLIB,DISP=SHR
```

The OMVS=xx parameter reflects which BPXPRMxx member OpenEdition MVS uses from SYS1.PARMLIB. The initialization parameters in BPXPRMxx define OpenEdition MVS processing controls and specify the name of the root file system to be used. F.2, "OpenEdition MVS Sample BPXPRMxx Parmlib Member" on page 212 shows an example of a BPXPRMxx member.

10.1.11 Making TSO/E Commands Available to ISPF Users

In order to make the OpenEdition MVS TSO/E commands available to ISPF users, you must concatenate the following target libraries to the appropriate ISPF data definition names:

- your_inst_hlq.SBPXEXEC concatenated to SYSEXEC or SYSPROC
- your_inst_hlq.SBPXTENU concatenated to ISPTLIB
- your_inst_hlq.SBPXPENU concatenated to ISPPLIB
- your_inst_hlq.SBPXMENU concatenated to ISPMLIB

You may also allocate the OpenEdition libraries using LIBDEFs.

Since many TSO/E users prefer to use ISPF selection panels, you may want to add these OpenEdition MVS commands to a panel, as shown in F.3, "OpenEdition MVS Sample ISPF Selection Panel" on page 213.

When ISPF customization is complete, OpenEdition MVS can be started by entering the MVS command S OMVS. Successful initialization appears on the console or system log as:

```
S OMVS
$HASP100 OMVS     ON STCINRDR
IEF695I START OMVS WITH JOBNAME OMVS IS ASSIGNED TO USER OMVSKERN
 , GROUP SYS1
$HASP373 OMVS     STARTED
BPXF013I FILE SYSTEM OMVS.ROOT.HFS 831
WAS SUCCESSFULLY MOUNTED.
BPXF203I DOMAIN AF_INET WAS SUCCESSFULLY ACTIVATED.
BPXF203I DOMAIN AF_UNIX WAS SUCCESSFULLY ACTIVATED.
BPXI004I OMVS INITIALIZATION COMPLETE
```

10.1.12 Customizing the OpenEdition MVS Root File System

Once OpenEdition MVS has been initialized, the root file system can be prepared for use by users and other OpenEdition MVS components.

The BPXISMKD member of SYS1.SAMPLIB, contains a supplied REXX EXEC that can be used to create the required directories for the root file system. This EXEC also creates the necessary special character files required by OpenEdition MVS. BPXISMKD should be run by a user ID defined as a *superuser*, and may

need to be customized to match the values selected in the BPXPRMxx member of SYS1.PARMLIB.

10.1.13 Initial Verification Checks

The following steps help to determine if OpenEdition MVS is functioning correctly.

1. Enter the MVS command D OMVS,A=ALL as shown below:

```
D OMVS,A=ALL
BPXO001I 15.39.57 DISPLAY OMVS 336
OMVS     ACTIVE                                        BPXPRM00
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
OMVSKERN BPXYAPPC 0022          1          0 1WI   09.29.16       .039
```

   Note: OMVSKERN is the user ID associated with the OMVS started procedure, and BPXYAPPC is the executing task.

2. Select the **1F** and **2F** options from your customized ISPF selection panel, or enter the OBROWSE and OEDIT commands from TSO/E.

   Successful results should be:

```
BPXWBBR--------------------  BROWSE - ENTRY PANEL  --------------------
Command ===>


Directory      ===> .


Filename       ===>

If file is to be browsed as fixed length records:
 Record length ===>         Leave blank to browse delimited text files


F1=HELP    F2=SPLIT    F3=END      F4=RETURN    F5=RFIND     F6=RCH
F7=UP      F8=DOWN     F9=SWAP     F10=LEFT     F11=RIGHT    F12=RET
```

```
BPXWEED--------------------   EDIT  - ENTRY PANEL  --------------------
Command ===>


Directory      ===> .


Filename       ===>

Profile name   ===>

Initial macro  ===>



F1=HELP    F2=SPLIT    F3=END      F4=RETURN    F5=RFIND     F6=RCH
F7=UP      F8=DOWN     F9=SWAP     F10=LEFT     F11=RIGHT    F12=RET
```

3. Select option **ISH** from your customized ISPF selection panel.

```
  File  Directory  Special_file  File_systems  Options  Setup  Help
--------------------------------------------------------------------------
BPXWP99                    OpenMVS ISPF Shell

Enter a pathname and press Enter or select an action bar choice.  You
can also specify an action code on the command line.

A blank pathname defaults to the pathname of your working directory.

Return to this panel to work with a different pathname.
                                                                  More: +
   /
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________


(C) Copyright IBM Corp., 1993.  All rights reserved.
Command ===> ____________________________________________________________
 F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
F10=Actions  F11=Command  F12=Cancel
```

All the functions of OpenEdition MVS may now be used via the ISPF shell interface and TSO/E commands. At the top of the ISPF shell panel you will see the pull-down menu selections that provide OpenEdition MVS functions. The detailed description of the ISPF shell can be found in *MVS/ESA OpenEdition MVS User's Guide SP V5*.

10.2 Installing Other OpenEdition MVS Components

During this project, the remaining OpenEdition MVS components were installed.

1. OpenEdition MVS application services
2. OpenEdition MVS shell and utilities
3. OpenEdition MVS dbx debugger

10.2.1 Application Services

OpenEdition MVS application services is a new component and consists of *tools* used in an application development environment. These tools are installed into both the root file system and some MVS data sets. They include items such as header files and socket utilities.

If you have installed OpenEdition MVS application services into different libraries than the ones used by AD/Cycle LE/370 1.3, you have to run a job to copy the necessary OpenEdition MVS elements into the LE/370 libraries. A sample job is provided in SYS1.SAMPLIB as member FOMISCPY.

10.2.2 Shell and Utilities

One of the user benefits of OpenEdition MVS is the OpenEdition MVS shell. The OpenEdition MVS shell is a command processor that you use to:

- Invoke shell commands or utilities that request services from the system (similar to TSO/E)
- Write shell scripts using the shell programming language (similar to REXX)
- Run shell scripts and C-Language programs interactively (in the foreground), in the background or in batch (again, similar to REXX)

10.2.3 Invoking The Shell

Once the OpenEdition MVS shell and utilities features have been installed, it can be invoked through a new TSO/E command OMVS. Like the OBROWSE, OEDIT, and ISHELL commands, the OMVS command can also be added to an ISPF selection panel, or entered as a TSO/E command.

As a quick way into the OpenEdition MVS shell, a user profile can also be customized to invoke the OMVS command at TSO/E logon time. This results in the OpenEdition MVS shell being presented to the user instead of the normal ISPF primary selection panel.

After the OMVS command is entered, the system initializes the shell for that user.

If successful, the user is presented with:

```
IBM
Licensed Material - Property of IBM
5655-068 (C) Copyright IBM Corp. 1993
(C) Copyright Mortice Kern Systems, Inc., 1985, 1993.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

# 1




 ===>                                                          INPUT
   2
ESC=¢ 1=Help    2=SubCmd 3=HlpRetrn  4=Top      5=Bottom   6=TSO
      7=BackScr 8=Scroll 9=NextSess 10=Refresh 11=FwdRetr 12=Retrieve
```

1 is the character used as the shell prompt, and **2** is the special character to escape from certain routines. These special characters may differ from country to country depending on the code page used to interpret characters.

10.2.4 Verifying The Shell

Note: For those users unfamiliar with the OpenEdition MVS shell, the commands are case sensitive. *All* commands should be entered in lowercase. Uppercase causes the command to be *not found*, and an *FSUM7351* message appears.

When the OpenEdition MVS shell has been successfully initialized, you can use any of the OpenEdition MVS commands documented in *MVS/ESA OpenEdition MVS Command Reference*. A few examples have been included here to help perform some simple verification checks.

1. Enter pwd

This results in OpenEdition MVS returning to the user the path of the present working directory. This should be the root directory, which is represented by a */*.

2. Next try env

   OpenEdition MVS returns the shell environment variables that have been used to initialize this OpenEdition MVS shell.

```
PATH=:/bin
SHELL=/bin/sh
COLUMNS=80
_=/bin/env
LOGNAME=userid
TERM=dumb
HOME=/
LINES=20
TZ=UTC0
```

3. Enter `ls -l`

   This results in OpenEdition MVS returning a list of all the files within the present working directory and some specific file information.

```
drwxr-xr-x   3 IBMUSER  SYS1           0 Jun  6 18:49 bin
drwxr-xr-x   2 IBMUSER  SYS1           0 May 27 13:21 dev
drwxr-xr-x   4 IBMUSER  SYS1           0 Jun 28 20:02 etc
drwxr-xr-x   2 IBMUSER  SYS1           0 May 27 13:20 lib
drwxrwxrwx   3 IBMUSER  SYS1           0 Jun 28 20:29 tmp
drwxr-xr-x   6 IBMUSER  SYS1           0 Jun 28 22:16 u
drwxr-xr-x   6 IBMUSER  SYS1           0 May 27 13:20 usr
```

4. Enter `c89 -o hfsinfo /etc/samples/hfsinfo.c`

   This causes a new executable program hfsinfo to be created in the present working directory. If successful, the shell returns only the prompt.

5. Enter `.hfsinfo` (Period is required)

   The program, hfsinfo, executes and displays information about the HFS. It should look similar to:

```
================================================
number of directories             = 33
number of regular files           = 290
number of FIFO files              = 0
number of symbolic links          = 0
number of character special files = 581
number of block special files     = 0

Total number of bytes             = 21371909
================================================
```

6. Enter `find / -name hobbies`

   This causes OpenEdition MVS to search all of the directories in the HFS, beginning at the root directory, and display the answer on the screen. The resulting display shows the full pathname for each occurrence of the file "hobbies."

```
/etc/samples/hobbies
```

10.2.5 dbx Debugger

Installation of the dbx component of OpenEdition MVS is done though SMP/E. During the installation process, data and files are loaded directly into the root file system. Therefore, OpenEdition MVS must be active at the time. If you have installed the dbx utility with SMP/E 1.8.0, do not forget to run the FDBXUSCH job to permit access to the dbx files (This is not required if you have installed with SMP/E 1.8.1).

Once installed, the dbx component can be verified by entering the OpenEdition MVS command dbx in an active OpenEdition MVS shell. If the installation has been successful, OpenEdition MVS prompts the user, as shown here:

```
dbx for MVS.
Type 'help' for help.
enter object file name (default is a.out', ¬D to exit):
```

If you do not have a program written in C available at this time, you can end the dbx session by entering the escape character, ¢, and d or the subcommand quit.

10.2.6 The OHELP Facility

OpenEdition MVS provides a help function available from the OpenEdition MVS shell through the OHELP TSO command. OHELP calls BookManager/Read and provides the user with online access to the necessary OpenEdition MVS books. To invoke the OHELP function, enter OHELP * followed by the OpenEdition MVS item you require, and press PF6. Bookmanager/Read is invoked and searches for the topic you entered.

```
OHELP * CEE3102
```

Bookmanager/Read displays the information for the LE/370 run time message CEE3102.

The following books are listed in the OHELP clist:

- *MVS/ESA OpenEdition MVS Command Reference*
- *IBM SAA AD/Cycle C/370 Library Reference*
- *MVS/ESA OpenEdition MVS Shell and Debugger Messages*
- *MVS/ESA Application Development Reference: Assembler Callable Services for OpenEdition MVS*

We recommend that you add extra books to this bookshelf to provide additional OpenEdition MVS help information to your users:

- *IBM SAA AD/Cycle Language Environment/370 Debugging Guide and Run-Time Messages*
- *IBM SAA AD/Cycle C/370 Library Reference: OpenEdition MVS Sockets*

- *MVS/ESA SP V5 OpenEdition MVS Users Guide*

To provide the OHELP facility, it is necessary to create data sets for the books and one for the OpenEdition MVS bookshelf. Further information on how to use and customize BookManager READ/MVS can be found in *BookManager READ/MVS Installation Planning and Customization* and *BookManager READ/MVS Getting Started and Command Summary* respectively.

A problem arises in that Bookmanager/Read uses the IBM C/370 runtime library; whereas OpenEdition MVS needs the IBM AD/Cycle LE/370 runtime library. To allow both Bookmanager and OpenEdition MVS OHELP to function correctly, you have to have the IBM C/370 library ahead of the LE/370 library in the linklist concatenation.

10.2.7 Batch Facilities

OpenEdition MVS provides users with a batch utility, BPXBATCH, that can be used to invoke OpenEdition MVS facilities in a batch environment.

The executable program BPXBATCH is called in the same way as any other MVS program, but because of the nature of OpenEdition MVS, there are some unique parameters and DD cards.

```
//BPXBATCH JOB (POK,999),MARTIN,MSGLEVEL=(1,1),MSGCLASS=X,
//  CLASS=A,NOTIFY=MARTIN
//*
//BPXBATCH  EXEC PGM=BPXBATCH,PARM='sh ls -al'
//STDOUT    DD PATH='/u/martin/stdout',
//             PATHOPTS=(OWRONLY,OCREAT),
//             PATHMODE=SIRWXU
//STDERR    DD PATH='/u/martin/stderr',
//             PATHOPTS=(OWRONLY,OCREAT),
//             PATHMODE=SIRWXU
```

In the above example, BPXBATCH is used to execute the OpenEdition MVS shell command `ls -al`. The output of this command is written to a file named STDOUT in the HFS. The file stdout is in the directory named /u/martin. Note that this HFS pathname is in lowercase. If any error messages are created, they are written to a different file, STDERR, in the same HFS directory as STDOUT.

An alternative means of passing the shell script information is by omitting the PARM='SH...', since this is the default, and instead adding a STDIN DD card. This example shows that the OpenEdition MVS shell commands to be executed can be found in the file 'Batch.in', which is also in directory name /u/martin.

```
//BPXBATCH JOB (POK,999),MARTIN,MSGLEVEL=(1,1),MSGCLASS=X,
//  CLASS=A,NOTIFY=MARTIN
//*
//BPXBATCH  EXEC PGM=BPXBATCH
//STDIN     DD PATH='/u/martin/Batch.in',PATHOPTS(ORDONLY)
//STDOUT    DD PATH='/u/martin/stdout',
//             PATHOPTS=(OWRONLY,OCREAT),
//             PATHMODE=SIRWXU
//STDERR    DD PATH='/u/martin/stderr',
//             PATHOPTS=(OWRONLY,OCREAT),
//             PATHMODE=SIRWXU
```

Since files and directory path names may be a mixture of uppercase and lowercase, it is important to ensure you have *exactly* the correct text when using BPXBATCH.

The parameters used on the BPXBATCH DD cards are explained in *MVS/ESA OpenEdition MVS Users Guide*.

The BPXBATCH facility can be used via a REXX EXEC named OSHELL. Rather than code a member of a PDS or PDSE to run BPXBATCH, you can simply run OSHELL from TSO and pass OpenEdition MVS commands as a parameter. For example:

```
oshell ls -al
```

Note: However, since ISPF always converts the lowercase characters to uppercase, the OpenEdition MVS command fails. This oshell EXEC can only be used at the TSO READY prompt.

10.2.8 Other TSO Commands

There are a number of special OpenEdition MVS TSO commands that can be used to move data between HFS data sets and traditional MVS data sets.

OCOPY Copy an MVS data set member into an HFS file, using DD names. Copy an HFS file into a MVS data set member, using DD names. Copy an HFS file into another HFS file. Copy an MVS data set member into another MVS data set member.

OGET Copy an HFS file into an MVS sequential data set, or partitioned data set member.

OGETX Copy one or more HFS files to a partitioned data set, a PDS/E or a sequential data set.

OPUT Copy an MVS sequential data set or partitioned data set member into an HFS file.

OPUTX Copy one or more members of a partitioned data set, PDS/E or a sequential data set to a directory.

A full description of these special TSO commands can be found in *MVS/ESA OpenEdition Command Reference*.

10.3 OpenEdition MVS Enhancements in MVS/ESA SP 5.1.0

MVS/ESA SP 5.1.0 introduces new functions and facilities to the OpenEdition MVS user:

- Multiple shell sessions
- dbx debugger enhancements
- NFSS support
- Internationalization
- Integrated sockets
- RMF enhancements
- REXX support

Each of these items is covered with emphasis on implementation and customization for your environment.

10.3.1 Multiple Shell Sessions

OpenEdition MVS provides a user with the ability to run multiple OpenEdition MVS shells from his 3270 terminal. Multiple OpenEdition MVS shells can be established at shell initialization time or after a single shell has been created. The creation of multiple shells at shell initialization time is accomplished through the use of a parameter, sessions(n) where n is the number of sessions you want, appended to the TSO OMVS command.

For example, the command OMVS SESSIONS(4) causes OpenEdition MVS to initialize four shells.

To create an additional shell if only one shell was initially created, use the new OpenEdition MVS subcommand open.

Whenever more than one shell has been created, it can be identified by a shell ID **1** in the lower right corner of the screen.

```
===>                                                              1
                                                                INPUT<3>
ESC=¢ 1=Help    2=SubCmd 3=HlpRetrn   4=Top       5=Bottom    6=TSO
      7=BackScr 8=Scroll 9=NextSess  10=Refresh  11=FwdRetr  12=Retreive
```

Moving between the sessions can be done with PF keys, or OpenEdition MVS subcommands. The new OpenEdition MVS subcommands are:

- OPEN
- CLOSE
- QUITALL
- NEXTSESS
- PREVSESS

To close each shell independently, use either the shell and utilities command exit, or the OpenEdition MVS subcommand quit. To close all the active shells at once, use the OpenEdition MVS subcommand quitall.

You may wish to add this option to your own ISPF environment.

```
OS,'CMD(OMVS SESSIONS(&CMDPRM))'
O4,'CMD(OMVS SESSIONS(4))'
```

With ISPF/PDF V3.5, entering OS causes a prompt for the number of OpenEdition MVS shell sessions required:

```
IKJ56700A ENTER Number of initial shell sessions to start -
```

If you have IPSF V4, entering the option OS *n* automatically creates *n* number of active shell sessions.

For more information on the use of the multiple sessions option, see *MVS/ESA OpenEdition MVS User's Guide MVS/ESA SP V5*.

10.3.2 dbx Enhancements

The dbx debugger is a utility that provides a symbolic debug program for C programs, allowing you to:

- Examine object files
- Provide a controlled environment for running a program
- Debug using symbolic variables and display them

In OpenEdition MVS, the dbx debugger now allows users to debug multi-threaded applications at the C source or at machine level. This is achieved by expanding both the current dbx command language and its processing, to be able to handle the new objects: threads, mutexes, and condition variables.

These new facilities are documented in the *MVS/ESA OpenEdition MVS Command Reference MVS/ESA SP V5*.

10.4 Internationalization

One of the major problems with software production is the hard-coding of language and cultural specifics in a program. As standards emerge and software and hardware vendors produce products for use in more than one country, it becomes critical for operating systems to be internationalized. The main advantage of internationalization is that a program can run in different language environments, or “locales” without being changed.

Internationalization has different aspects:

Viewed from the installation: setting the values for shell variables, code pages and so forth, correctly according to nationalities. One problem that is observed in many installations is described more detailed in 10.4.3, “Square Bracket Solution” on page 179.

Seen by the application programmer: how can we simply implement “national” behavior of an application. For example, how can we display date and time according to nationalities without a large overhead in coding.

In OpenEdition MVS, you can set the locale for your C code applications by using the setlocale() function, or if you are within the shell you can set it to an environment variable. If an application, be it a C program or a shell script, is started, the current settings of environment variables may be inherited.

10.4.1 Locale Definitions

The following flavors can be adapted according to current national environment.

LC_COLLATE Defines the collating sequence of elements in the programs locale, commonly used by the sort function.

LC_CTYPE Defines character classification and case conversion for characters in the programs locale, that is, case sensitivity and digits and so forth.

LC_MESSAGES Defines the format and values for positive and negative responses.

LC_MONETARY Defines the rules and symbols used to format monetary numeric information.

LC_NUMERIC Defines the rules and symbols used to format nonmonetary numeric information, namely decimal separator, thousand separator and so forth.

LC_TIME Defines the time and date formats, the order of the fields, numbers or letters, 12 hour clock or 24 hour clock.

LC_TOD Defines time zone and daylight savings information.

LC_ALL Defines all of the above.

These variables can be set in either the *shell* or in a POSIX application. For example, if you set:

```
LC_ALL='De_DE.IBM-1047'
```

LC_ sensitive commands reply according to the national behavior of Germany:

```
pwd=/u/hdm: >echo $LC_ALL
De_DE.IBM-1047
pwd=/u/hdm: >date
Di  5 Jul 21:58:20 1994
pwd=/u/hdm: >
```

OpenEdition MVS supports both single- and double-byte character set code pages.

10.4.2 The Internationalization API

The implementation of POSIX provides a large amount of functions that take care of nationalities. A simple example is shown in F.6, “Internationalization” on page 218. There we use the time formatting routine `strftime()` to format date and time. `strftime()` *implicitly* looks for the country specific behavior. Our program displays (see Figure 74 on page 179) the current setting of the system variables defining the national behavior **1**. It then displays date and time according to these current settings. After changing the behavior to a different country **2**, in our case GERMany, date and time are now displayed according to the new settings.

Note: Our program was called from the shell. The settings as defined in that environment **1** were “exported” to our program. The installation has to take care of the correct settings according to the user’s country.

```
pwd=/u/hdm/omvsc: >country GERM

The original values were...

Current time zone setting : UTC0          1
LC_COLLATE:  C
LC_CTYPE:    C
LC_MONETARY: C
LC_NUMERIC:  C
LC_TIME:     C
LC_TOD:      C

06/22/94 23:30:54                 1  2
Parameter received germ

Going to change locales according to GERM

Current time zone setting : MEZ-1MESZ-2   1
LC_COLLATE:  GERM
LC_CTYPE:    GERM
LC_MONETARY: GERM
LC_NUMERIC:  GERM
LC_TIME:     GERM
LC_TOD:      GERM

23.06.1994 01:30:54               1  2
```

Figure 74. Internationalization

10.4.3 Square Bracket Solution

As shown in Figure 75, the translation and mapping of the square brackets within code page 0037 does not conform to the mapping that is defined in code page 1047: the brackets are located on positions X'AD' and X'BD' in code page 1047 and are mapped to the ´ (Y acute) and the ¨ (umlaut) in code page 0037. The actual translation into the character actually displayed is done at the workstation according to the setting of code pages. In our implementation we used a PS/2 running OS/2 2.1 and CM/2 1.1.

```
pwd=/: >date -g
FSUM9380 date: Unknown option rcv -g
FSUM9299 Usage: date ·-cu¨ ·+format¨
 date ·-cut¨ <date_time>
pwd=/: >
```

Figure 75. Square Bracket with Code Page 0037

Note: Though we show in Figure 75 the ¨ it was actually not shown on our system due to the final translation from 37 to 437. In fact, the ″ (quote mark) was shown.

The OMVS command sends the 3270 data stream according to code page 1047 to the workstation, so the final translation has to be corrected. We did this in a way described detailed in *Communications Manager/2 Workstation Installation and Configuration Guide*. We provide a short overview of the process, but we recommend having the cited book available when actually changing the translation tables.

1. Extract the code pages using ACSGCCRT.
2. Copy the code page 037 contained in file 037.cpt as a “user code page.” A user code page must have a number in the range from 65280 to 65534.
3. The square brackets are located on X′AD′ and X′BD′ corresponding to lines 174 and 191 respectively. The translations are defined symbolically, that means “Right_Bracket” designates the “]” character.
4. We replaced current translations of [and] located at X′BA′ and X′BB′ into “Undefined.” The translations in line 174 and 191 were replaced with the symbolic names for the brackets.

After these actions, the square brackets were displayed correctly, as you can see in Figure 76.

```
pwd=/u/hdm: >date -g
FSUM9380 date: Unknown option rcv -g
FSUM9299 Usage: date [-cu] [+format]
 date [-cut] <date_time>
pwd=/u/hdm: >
```

Figure 76. Square Bracket with a Modified 037 Code Page

Note: In case your reading environment has to deal with code pages other the 0037, we strongly recommend reading *AD/CYCLE C/370 Release 2 Programming Guide for Language Environment/370 Release 3*.

10.4.4 Variant Characters

The POSIX Portable Character Set (PPCS) identifies the core set of 128 characters that are needed to write code and run applications. Of these, 13 characters are variant among the EBCDIC coded character sets. Table 7 lists these 13 characters and shows how the character mapping in the code pages is actually done.

Table 7. Mappings of the 13 PPCS Variant Characters

Character	Open Systems Hex Value (Default)	Open System IBM-1047 view	APL IBM-293 view	Inter-national IBM-500 view	France IBM-297 view	Germany IBM-273 view	US/Can IBM-0037 view
left bracket	AD	[	]	.	.	.	.
right bracket	BD	]	[	ü	~	ü	¨
left brace	C0	{	{	{	é	ä	{
right brace	D0	}	}	}	è	ü	}
backslash	E0				ç	Ö	
circumflex	5F	^	¬	^	^	^	¬
tilde	A1	~	~	~	ü	ß	~
exclamation mark	5A	!	!	]	§	Ü	!
pound (number sign)	7B	#	#	#	£	#	#
vertical bar	4F	|	|	!	!	!	|
accent grave	79	`	`	`	µ	`	`
dollar sign	5B	$	$	$	$	$	$
commercial “at”	7C	@	@	@	á	§	@

In an environment working with a codepage mentioned in Table 7, special care must be taken when copying data containing source type data into the HFS.

10.4.5 Copying Data from MVS Data Sets

There are several new TSO/E commands provided by OpenEdition MVS (see 10.2.8, “Other TSO Commands” on page 175) that allow you to copy data located in MVS data sets to the hierarchical file system and vice versa. All of these can convert during copying.

```
OPUT 'A2.C(TEST)' '/u/hdm/omvsc/test.c' convert(BPXFX111)
```

In this example, the C source code is copied to the HFS. BPXFX111 specifies a non-APL conversion table to convert from IBM-037 to IBM-1047.

10.4.6 Exchanging Data with Workstations

OpenEdition provides MVS with a view that is very similar to that of a workstation as seen from the file structure and organization of files.

A very popular file packaging technique used on UNIX systems is the *tar* file (tape archive). This type of file can be handled within the shell through either the `tar` or `pax` command. `pax` is an additional shell command that can be used to handle this kind of archive. The advantage of `pax` over `tar` is that one can select a code page for translation. When using archive files:

```
pax -wf archive.pax -o from=IBM-1047,to=IS646  /u/hdm/omvsc
```

With this command, files from the directory /u/hdm/omvsc are stored in a file archive.pax. While creating the file, a code conversion is done from code page 1047 to an ASCII character set (IS646).

10.4.7 Network File System Support

Network File System(NFS) support is a means to access data on a remote computer without actually copying the data to the local system. NFS provides the user with *online file sharing*. It was developed by SUN Microsystems Inc. and published in Internet RFC 1094. NFS server support is implemented in MVS as a separate feature of DFSMS. A principle scheme is shown in Figure 77 on page 182. NFS provides access to the hierarchical file system. By using the hierarchical file system, NFS and OpenEdition MVS gives the user the same view into files and their structure as an OS/2 or AIX server does. Travelling through directories located in the hierarchical file system is the same as on the workstation.

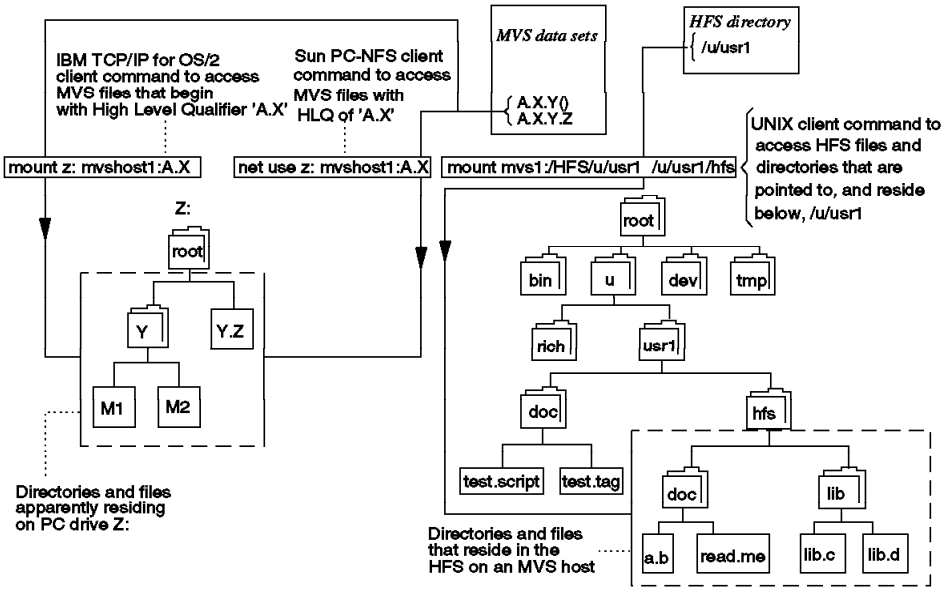


Figure 77. File Access Using NFS

A simple example is given in Figure 78 on page 183. After an initial login **1** to the file server, the user gains access to the remote data he wants to share through a *mount* **2** command. For the example shown here, an OS/2 workstation was used. To access remote files from OS/2, a drive letter must be assigned. This is done through the `mount` command. After having assigned a drive letter, data can be easily accessed. At **3** we use a `directory` command to show the contents of the current directory. At **4** a `logoff` is done.

```
[R:\]mvslogin c1jes2 hdm                1
Password required
GFSA973A Enter MVS password:
GFSA955I hdm logged in ok.

[H:\SCRIPT]mount r: c1jes2:/HFS/u/hdm     2
mount: c1jes2:/HFS/u/hdm

NFS Drive 'r:' was attached successfully.

[H:\SCRIPT]r:

[R:\]dir                                  3

 The volume label in drive R is NFS.
 The Volume Serial Number is 001E:0000
 Directory of R:\

 7-05-94   4:15p      1229           0  .profile
 7-05-94   3:47p       146           0  .setup
 7-06-94   5:48p      2131           0  .sh_history
 7-06-94   4:58p     <DIR>           0  omvsc
 6-23-94  10:16a      3997           0  pconway.sys
 6-27-94   7:46p         0           0  rconway.sys
        6 file(s)       7503 bytes used
                        8192 bytes free

[R:\]mvslogut c1jes2                      4
GFSA958I uid 0 logged out ok.
```

Figure 78. HFS Access from an OS/2 System

For comparison we show in Figure 79, the output of the `ls` command done on the MVS host.

```
pwd=/u/hdm: >ls -l
total 16
drwxrwxrwx   2 IBMUSER  SYS1           0 Jul  6 22:58 omvsc
-r-x--x--x   1 IBMUSER  SYS1        3997 Jun 23 16:16 pconway.sys
-rwx--x--x   1 IBMUSER  SYS1           0 Jun 28 01:46 rconway.sys
-r-x--x--x   1 IBMUSER  SYS1        3997 Jun 23 16:16 zconway.liste2
pwd=/u/hdm: >
```

Figure 79. Directory Contents as Seen within OpenEdition MVS

10.4.8 Double-Byte Character Support (DBCS)

The DBCS support in OpenEdition MVS provides Japanese message translation for shell and utilities messages, OpenEdition MVS TSO commands and panels, and kernel write-to-operator messages.

OpenEdition MVS does not address internationalization support in the dbx debugger. The dbx debugger provides a minimum level of support for recognition, displaying, modifying and referencing of program variables and constants that may include DBCS characters.

10.4.9 Integrated Sockets

OpenEdition MVS solves a problem associated with issuing I/O to both files and sockets from the same program. Changes to TCP/IP, C/370 run time library and OpenEdition MVS ensure that there is only one version of the I/O function calls capable of handling both files and sockets.

The logical file system routes the request to the appropriate physical file system, HFS or TCP/IP.

OpenEdition MVS supplies all the header files required for using sockets. To use the IBM AD/Cycle C/370 compiler in batch, you have to update the EDCCPLG cataloged procedure to include the OpenEdition MVS header file data set.

1. After the //SYSLIB DD statement, add the following line:

   ```
   //   DD     DSN=_your_hlq_.SFOMHDRS,DISP=SHR
   ```

2. In the compile step, change the CPARM parameters to:

   ```
   CPARM='DEF(MVS,_OPEN_SOCKETS,_POSIX1_SOURCE=1),RENT,LO'
   ```

3. For the prelink edit step, add the following prelink parameter:

   ```
   PPARM='OMVS',
   ```

It is more likely that you will use the OpenEdition MVS c89 utility to compile and link-edit your programs. You must use the following options on the c89 command:

- -D MVS
- -D _OPEN_SOCKETS
- -I "//'_your_hlq.SFOMHDRS'"

The option indicators "-D" and "-I" must be in *uppercase* as the defines (-D) must be. The actual command looks like:

```
c89 -I "//'OMVS.SFOMHDRS'" -D MVS -D _OPEN_SOCKETS -o mod mod.c
```

Where "mod" is the name of the executable module, and "mod.c" is the name of the source file, which may well be a lower-, upper- or mixed-case name.

A simple socket application, consisting of server and client programs, can be used to show integrated sockets. The server program opens a socket in one shell session and waits for the client to connect to it. The client then sends a predetermined amount of data and this is returned by the server. The sample programs can be found in F.5, "OpenEdition MVS Sample Socket Application" on page 216.

```
pwd=/: >server 2001 16
socket() returned 3
bind() returned 0
Server listens on port 2001
listen returned 0
blocking on accept() for an incoming connection ..
```

The client was started in another session:

```
pwd=/: >client 9.12.13.69 2001 16 5
socket() returned 3
Establised connection with system 0x90c0d45
Waiting 5 seconds before sending data ...
```

The server receives the data from the client, returns it and closes the socket.

```
Established connection with system 0x90c0d45
Now attempting to receive data ...
recv() returned 16 bytes
All data transferred closing socket...
pwd=/: >
```

The client receives the returned data and shuts down.

```
Now attempting to receive data ...
recv() returned 16 bytes
recvbuf contains:
abcdefghi«»‰yû·°
pwd=/: >
```

For more information on the use of OpenEdition MVS sockets, see *IBM SAA AD/Cycle C/370 Release 2 Library Reference: OpenEdition MVS Sockets*.

10.4.10 SMF Accounting

There are unique functions in OpenEdition MVS that need to be considered when doing accounting. The “exec” family of functions causes step termination and a new substep to be started. However, this new substep still has the same step number, but the substep number is incremented. Therefore, accounting applications must look for substep_number in addition to step_number, jobname and job_start_time.

There is a new macro, BPXESMF, to assist installations in collecting accounting information from OpenEdition MVS.

See the IBM publication *MVS/ESA SPV5.1 Systems Modification Facility*.

10.4.11 Using RMF V5

Resource Measurement Facility (RMF) Version 5 provides OpenEdition MVS report information using existing RMF records together with some new ones:

- Record type 42 subtype 6 provides information about data set level performance.
- Record type 74 subtype 3 provides data on kernel activity, numbers of users, processes and so forth.

- Record type 92 provides information about the file system activity, mount, demount, open, close, suspend and resume.

A new RMF report, OMVS kernel activity report, provides information on:

- OMVS system call activity
- OMVS process activity

The reporting options that need to be selected for OpenEdition MVS are shown in *RMF V5.1 Users Guide*.

10.4.12 RMF Monitor III

RMF invokes the Monitor III procedure *RMFGAT* to obtain OpenEdition MVS data. RMFGAT must be defined or defaulted to having a user ID of RMFGAT, and this user ID must then be defined to RACF with the appropriate OpenEdition MVS authorization:

```
ADDUSER RMFGAT DFLTGRP(OMVSGRP) OMVS(UID(123) HOME('/'))
```

If you do not define RMFGAT in this way and RMF is started automatically by the system, you receive ICH408I messages when OpenEdition MVS is initialized.

Monitor III data gatherer collects OpenEdition MVS data for input to the RMF post processor. This data can then be used to create the OMVS kernel activity report. Figure 80 shows an example of this RMF report.

```
                                    O M V S   K E R N E L   A C T I V I T Y
                                                                                                                                                PAGE  1
           MVS/ESA                 SYSTEM ID 9121            START 06/22/94-13.42.53  INTERVAL 000.15.47
           SP5.1.0                 RPT VERSION 5.1.0         END   06/22/94-13.58.40  CYCLE 1.000 SECONDS
TOTAL SAMPLES =   946
                                                 OMVS SYSTEM CALL ACTIVITY
-------------------------------------------------------------------------------------------------------------------------------------------------------
                MINIMUM  AVERAGE  MAXIMUM
-------------------------------------------------------------------------------------------------------------------------------------------------------
 SYSCALLS (N/S)   1.000    4.424    247.0
 CPU TIME (H/S)   0.000    0.005    1.000
                                                   OMVS PROCESS ACTIVITY
-------------------------------------------------------------------------------------------------------------------------------------------------------
                           PROCESSES                        USERS                    PROCESSES PER USERS
 MAXIMUM  (TOT)              300                              200                          25
-------------------------------------------------------------------------------------------------------------------------------------------------------
                MINIMUM  AVERAGE  MAXIMUM      MINIMUM  AVERAGE  MAXIMUM      MINIMUM  AVERAGE  MAXIMUM
-------------------------------------------------------------------------------------------------------------------------------------------------------
 CURRENT  (TOT)       5    5.852        8            1    1.000        1
 OVERRUNS (N/S)   0.000    0.000    0.000        0.000    0.000    0.000        0.000    0.000    0.000
 Units:   (TOT) = Total Value, (N/S) = Number per Second, (H/S) = Hundredths of seconds per Second
```

Figure 80. Sample OpenEdition MVS RMF Kernel Activity Report

For a detailed description of the RMF OMVS system activity report, please refer to *MVS/ESA Analyzing Resource Measurement Facility Version 5 Reports*.

10.4.13 REXX Programs

The set of OpenEdition MVS extensions to the TSO/E REXX language now enable REXX programs to access OpenEdition MVS callable services. The OpenEdition MVS extensions, called syscall commands, have names that correspond to the names of the callable services that they invoke, for example, *access*, *chmod*, and *chown*.

You can only run a REXX program with syscall commands on a system with OpenEdition MVS installed and from either, TSO/E, batch, the OpenEdition MVS shell or another program.

Two new host command environments are available, they are:

SYSCALL For a REXX program with syscall commands that runs from TSO/E or MVS batch. You need to initialize the environment by beginning a REXX program with a `syscalls('ON')` call.

SH For a program with SYSCALL commands that runs from a shell or a program, SH is the initial host environment. The SYSCALL environment is automatically initialized as well, so you do not need to begin the program with a `syscalls('ON')` call.

The `syscalls('ON')` establishes the SYSCALL environment and sets up predetermined variables, blocks signals and returns a return code. This return code may not always be zero, so you should always test for this value when establishing the syscalls environment. The example shows a simple REXX EXEC that tests the SYSCALLs return code and list information for the root directory.

```
/* REXX */
  if syscalls('ON')>0 then
    do
    say 'Unable to establish the SYSCALL environment'
    return
  end
 address syscall
 'readdir / root.'
 do i=1 to root.0
 say root.i
end
```

This example shows REXX code to be used within the OpenEdition MVS shell environment.

```
 /* REXX */
 parse arg dir 'ON'
 address syscall 'getcwd cwd'
 say 'current working directory is' cwd
 if dir='' then
    do
    say 'enter directory name to list'
    parse pull dir
    end
 'ls -al' dir
 return
```

These examples are taken from *Using REXX to Access OpenEdition MVS Services*.

10.4.14 Tuning OpenEdition MVS

There are a number of tuning activities that can enhance the performance of OpenEdition MVS.

Overall OpenEdition MVS performance can benefit from adding the following modules to the link pack area.

- BPXEV003, BPXPLPKA, BPXOLVD, BPXOV, BPXMMS, BPXUCSNM, BPX$LCNM.

These modules are loaded into ELPA, and their total storage requirement is approximately 1.5MB.

The performance characteristics of OpenEdition MVS can be set by customizing the IEAIPSxx and IEAICSxx members of SYS1.PARMLIB, or if you are using goal mode by defining OpenEdition MVS to the workLoad manager.

File system performance is dependant on how the file systems are organized and where they are placed on DASD. RMF can be used to help improve file system performance.

For more information on tuning OpenEdition MVS, see *MVS/ESA Planning: OpenEdition MVS SP V5*.

10.5 Customizing OpenEdition MVS

This section describes the functions required to customize OpenEdition MVS for your environment.

10.5.1 Multiple Users

In traditional TSO/ISPF environments, a new user is given his own profile and the ability to create data sets under a certain HLQ. In OpenEdition MVS, this practice can be accomplished by giving the user his own HFS and OpenEdition MVS profile. This and any other user HFS should then be mounted onto the root file system at the /u directory. This allows the root file system to be used in read-only mode, much the same way as a user has read access to traditional MVS system libraries. Figure 84 on page 215 shows a logical representation of user file systems mounted on the /u directory.

This practice allows each OpenEdition MVS shell user to use his own HFS as he pleases, without the risk of impacting any other OpenEdition MVS shell users. IBM recommends that you adopt this approach to managing multiple file systems.

10.5.2 Creating the User File Systems

A user HFS is created in exactly the same way as any other HFS. Choose a name that represents the user and a size that provides sufficient space for the user requirements. If more space is required, you may wish to create additional HFSs for a user and mount them as required.

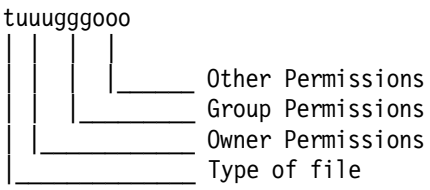
```
TSO ALLOCATE DATASET('MARTIN.HFS') DSNTYPE(HFS) SPACE(5,5) DIR(1) CYL
STORCLAS(smsclass)
```

Change to the /u directory, and issue the *mkdir username* command to create a directory entry. This entry is used as the mount point for the user HFS.

The new OpenEdition MVS shell and utilities user may require his HFS to be “protected” from other uses, as would be usual for traditional MVS data sets. In OpenEdition MVS this is done by setting *permission bits*. The permission technique is based on authority at three levels:

- The file OWNER
- The file owners GROUP
- All OTHER users

The permission bits are set in the same way for each level of access using a simple binary 10-bit code.



The type of file can be:

- - File
- c character special file
- d directory
- l symbolic link
- p FIFO special file

The next 3 bits represent the access control for each of the three groups, for example:

- Read,write,execute, (B′111′) is represented by X′7′
- Read,no write,no execute, (B′100′) by X′4′
- No read,no write,no execute, (B′000′) by X′0′

A user may set permission bits for any combination an any level of access. For example, If a user wishes to have, read, write and execute for his own files, but not allow access to any other member of his RACF defined default group, or any other OpenEdition MVS user, he would set the permission bits to *700*, which is displayed as: *rwx*------.

To set the permission bits for this new user HFS, issue the command:

```
chmod 700 /u/username.
```

Issue the command:

```
chown -R username /u/username
```

to set the owner of the mount point and HFS to the appropriate new username. Here is an example of three user directory entries in the /u directory, each with different permission bit settings, each a mount point for a user HFS.

```
drwxr-xr-x   3 MARTIN   SYS1          0 Jun  1 21:07 martin
drwxr-x---   3 HDM      SYS1          0 Jun  1 21:08 hdm
drwx--x--x   3 RICH     SYS1          0 Jun  3 12:45 rich
```

The user HFS that was previously created can now be mounted at the /u/username directory. The mount can be performed through:

- The TSO MOUNT command
- Using the ISPFShell mount options
- Adding an entry to the BPXPRMxx member in SYS1.PARMLIB and stopping/restarting OpenEdition MVS

An example of the TSO mount command would be:

```
mount filesystem('martin.hfs') type(hfs) mountpoint('/u/martin')
```

The OpenEdition MVS command df -P can be used to show all the file systems presently mounted.

```
Filesystem    512-blocks        Used  Available  Capacity Mounted on
MARTIN.HFS         12000        1972      10028       16% /u/martin
HDM.HFS             6000        1692       4308       28% /u/hdm
RICH.HFS           12000        3988       8012       33% /u/rich
OMVS.ROOT.HFS      60000       22320      37680       38% /
```

Each user who wishes to use OpenEdition MVS has to have an OpenEdition MVS segment in his RACF profile. This segment defines to OpenEdition MVS the access permissions of the user and a reference known as the UID. Use the RACF altuser command to define an OpenEdition MVS segment for each of your intended OpenEdition MVS users.

```
ALTUSER RICH OMVS(UID(nnn) HOME('/u/rich'))
```

The TSO user can now log on and called OpenEdition MVS from any of your chosen options.

10.5.3 The Default Profile

OpenEdition MVS uses a number of system wide variables for interaction with other IBM products and the OpenEdition MVS users. If your installation has used names for AD/Cycle C/370 and AD/Cycle LE other than:

- EDC.V1R2M0.xxxxxxxx or
- CEE.V1R3M0.xxxxxxxx

you need to customize the OpenEdition MVS profile to inform OpenEdition MVS of the new names.

The OpenEdition MVS system wide profile should be created at pathname */etc/profile*. You should customize this as shown here:

```
export_C89_CLIB_PREFIX="hlq.2ndlq"
export_C89_PLIB_PREFIX="hlq.2ndlq"
```

The complete list of OpenEdition MVS environment variables can be found in *MVS/ESA OpenEdition Command Reference*.

10.5.4 Creating User Profiles

OpenEdition MVS users may wish to have their own OpenEdition MVS profile. This can be accomplished by creating a file named *.profile* in their home directory. A sample profile is supplied in /etc/samples/.profile, and this can be copied and edited by using the OEDIT command. The following example shows a sample user profile that sets two OpenEdition MVS variables, PATH and PS1. In this example, the “path” statement sets the search order of directories to:

1. The present working directory **1**
2. The /bin directory **2**
3. The /u/hdm directory **3**

No other directories are searched. Other required directories can be added to the path statement as required.

```
      1  2    3
PATH='.:/bin:/u/hdm'
PS1='$LOGNAME:'$PWD':' >'
export PATH PS1
```

The PS1 statement sets the OpenEdition MVS shell prompt for this user to show the user login name, the pathname of the present working directory followed by the symbol >. This OpenEdition MVS shell prompt would appear for user HDM as:

```
HDM:/u/hdm: >
```

Once the new user file system has been mounted, the new user may login to OpenEdition MVS.

Figure 84 on page 215 shows the recommended approach to managing multiple HFSs in an OpenEdition MVS environment.

10.5.5 Customizing User PF Keys

OpenEdition MVS is initialized with a predefined set of PF keys. If you wish to change these definitions, you can do so by passing your own PF key definitions at OpenEdition MVS initialization time. To do this, create an OMVS REXX EXEC, as shown here:

```
/*  REXX  */
'OMVS PF4(CONTROL) PF10(PREVSESS)'
```

Instead of using the OMVS TSO command to initialize your shell and utilities session, execute your REXX EXEC.

On entering the OpenEdition MVS shell, your PF keys are displayed as:

```
===>
                                                              INPUT<4>
ESC=¢ 1=Help    2=SubCmd 3=HlpRetrn   4=Control   5=Bottom    6=TSO
      7=BackScr 8=Scroll 9=NextSess  10=Prevsess 11=FwdRetr  12=Retreive
```

In the example shown, setting PF4 to control can be particularly useful when using the OpenEdition MVS Shell and Utilities and the dbx debugger features. Setting PF10 to prevsess helps you to navigate around multiple shell sessions.

You may combine your PF key definitions and session parameters in the same REXX EXEC.

```
/*  REXX  */
'OMVS PF4(CONTROL) PF10(PREVSESS) SESSIONS(3)'
```

10.5.6 Backing Up and Restoring Files

There are no enhancements or changes to the way OpenEdition MVS can be used for backing up and restoring HFS files. Data Facility System Managed Storage Hierarchical Storage Manager (DFSMShsm) provides automatic facilities for backing up HFS data sets.

To back up an individual file, copy it into an MVS sequential data set, PDS or PDSE, and then use DFSMShsm to back up this data set.

10.5.7 File Transfer Program

One means by which existing data and programs can be brought into the OpenEdition MVS file system is through the use of File Transfer Program (FTP).

When a TCP/IP session can be established between the OpenEdition MVS host and a workstation, enter FTP *hostname* at the workstation. FTP then asks for logon information necessary to authorize the file transfer process. This information is host system user ID and password. Once verified, enter put or get depending on whether your are moving data to or from the host system. FTP then asks for input and output file names, and the transfer takes place.

If you are moving data up to the OpenEdition MVS host system, the output data set should be a sequential file. This can then be copied into the HFS by using the OpenEdition MVS TSO commands OGET, OCOPY or OEDIT.

FTP can be used as a quick means of transferring data to the OpenEdition MVS HFS without the need for tape or other external media processing.

For more information, see *IBM TCP/IP for MVS: User's Guide*.

Appendix A. A Sample Program Utilizing the IOSPTHV Macro

```
*********************************************************************
*     This SAMPLE program prompts the user for a device number      *
*     and returns path related information associated with the      *
*     device number.                                                *
*                                                                   *
*     The format of the prompt and an example of the output follows:*
*                                                                   *
*==> DEVICE NUMBER / END                                           *
*OFCO                                                               *
*                                                                   *
*DEVICE OFCO ONLINE         PATH VALIDATED                         *
*CHP              44 4C 34 5C                                      *
*CHP PHYS AVAILABE Y  N  N  Y                                      *
*CHP LOGI AVAILABE Y  N  N  Y                                      *
*DEVICE OFCO CHPID 44 VALIDATION RC/RSN 00000000/00000000 O.K.     *
*DEVICE OFCO CHPID 4C VALIDATION RC/RSN 00000004/00000004 PTH NOT AV *
*DEVICE OFCO CHPID 34 VALIDATION RC/RSN 00000004/00000004 PTH NOT AV *
*DEVICE OFCO CHPID 5C VALIDATION RC/RSN 00000000/00000000 O.K.     *
*********************************************************************
DEVACC   CSECT
DEVACC   AMODE 31                      31-BIT ADDRESSING MODE
DEVACC   RMODE 24                      RMODE ANY
         SPACE 1
*-------------------------------------------------------------------*
*         REGISTER ASSIGNMENTS                                      *
*-------------------------------------------------------------------*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6                       DYNAMIC AREA REGISTER
UCBPTR7  EQU   7                       UCB POINTER
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12                      MODULE BASE REGISTER
R13      EQU   13                      POINTER TO STANDARD SAVE AREA
R14      EQU   14
R15      EQU   15
*-------------------------------------------------------------------*
*        STANDARD ENTRY LINKAGE                                     *
*-------------------------------------------------------------------*
         PRINT GEN
         SAVE  (14,12),,&SYSDATE.=&SYSTIME.=DEVACC
         LR    R12,R15            ESTABLISH MODULE BASE REGISTER
         USING DEVACC,R12         SETS UP BASE REGISTER
         BAL   R14,SETSM
         MODESET KEY=ZERO,MODE=SUP
         LA    R0,DYNSIZE         LOAD LENGTH OF DYNAMIC AREA
         STORAGE OBTAIN,LENGTH=((R0)),SP=233 GETS DYNAMIC AREA
         LR    R6,R1              GETS DYNAMIC AREA ADDRESS
         USING DYNAREA,R6         SETS UP DYNAMIC AREA
         ST    R13,SAVE+4         SAVE CALLER'S SAVE AREA ADDRESS
         LA    R15,SAVE           GET THIS MODULE'S SAVE AREA ADDRESS
         ST    R15,8(R13)         SAVE THIS MODULES SAVE AREA ADDRESS
*                                 IN CALLER'S SAVE AREA.
         LR    R13,R15            SET UP ADDRESSABILITY TO THIS
*                                 MODULE'S SAVE AREA.
*-------------------------------------------------------------------*
*        ENSURE THAT THIS PROGRAM IS RUNNING ON AN MVS/ESA SP 5.1   *
*        SYSTEM BY CHECKING THE VERSION INFORMATION IN THE CVT.     *
*-------------------------------------------------------------------*
         L     10,X'10'(,0)       LOAD CVT POINTER
         USING CVT,10
         TM    CVTDCB,CVTOSEXT     IS THE OSLEVEL EXTENSION PRESENT
         BNO   NOSUPP              NO, PRE-MVS/SP VERSION 3 SYSTEM
         TM    CVTOSLV1,CVTH551O   RUNNING ON VERSION HBB551O?
         BNO   NOSUPP              NO, PRE-HBB551O SYSTEM. IOSPTHV
*                                  SUPPORTED ON HBB551O AND ABOVE
*-------------------------------------------------------------------*
*   SET UP ADDRESSABILITY TO A STORAGE AREA CALLED UCBSTOR INTO WHICH *
*   THE UCBSCAN MACRO WILL RETURN THE UCBS OF DEVICES IT LOCATES.   *
*-------------------------------------------------------------------*
         LA    UCBPTR7,UCBSTOR    GET ADDRESS OF WORK AREA
         USING UCB,UCBPTR7        SET UP ADDRESSABILITY
*-------------------------------------------------------------------*
*   CLEAR THE UCBSCAN WORK AREA.                                    *
*-------------------------------------------------------------------*
SCANLOOP DS    0H
         LA    R0,SCANWORK        SET STORAGE ADDRESS
         LA    R1,L'SCANWORK      SET STORAGE LENGTH
         SR    R15,R15            CLEAR SECOND OPERAND
         MVCL  R0,R14             CLEAR THE STORAGE
*-------------------------------------------------------------------*
*   PROMPT FOR DEVICE NUMBER.                                       *
*-------------------------------------------------------------------*
         LA    R1,=CL24'==> DEVICE NUMBER / END '
         LA    R0,24
         TPUT  (1),(0)            PROMPT
         MVC   DEVNUMC,=CL48' '
         LA    R1,DEVNUMC
         LA    R0,L'DEVNUMC
         TGET  (1),(0)            READ INPUT
         OC    DEVNUMC,=CL48' '   UPPER CASE
         LTR   R15,R15            TGET PROBLEMS?
         BNZ   SCANLOOP           YES - TRY AGAIN
         CLC   =C'END',DEVNUMC    EXIT REQUESTED?
         BE    EXIT               YES - EXIT
*-------------------------------------------------------------------*
*   PARSE INPUT                                                     *
*-------------------------------------------------------------------*
         LA    R0,L'DEVNUMC       PREPARE..
         LA    R1,DEVNUMC         .TO..
         SLR   R2,R2              ..SCAN..
         SLR   R3,R3              ...INPUT LOOP.
INPSCN   DS    0H
         CLI   0(R1),C' '         END OF INPUT?
         BE    INPDON             YES - EXIT LOOP
         TM    0(R1),X'F0'        DIGIT 0-9?
         BNO   INPCHA             NO - CHECK FOR A CHAR
         SLL   R2,4               APPEND..
         IC    R3,0(,R1)          .HEX..
         N     R3,=X'0000000F'    ..INPUT..
         OR    R2,R3              ...DIGIT
         LA    R1,1(,R1)          NEXT INPUT CHAR
         BCT   R0,INPSCN          LOOP IF NOT END OF INPUT
         B     INPDON             ALL INPUT DONE
INPCHA   DS    0H
         TM    0(R1),X'C0'        CHAR A-I?
         BNO   INPERR             NO - GIVE AN ERROR MESSAGE
         CLI   0(R1),C'F'         CHAR A-F?
         BH    INPERR             NO - GIVE AN ERROR MESSAGE
         SLL   R2,4               APPEND..
         IC    R3,0(,R1)          .HEX..
         N     R3,=X'0000000F'    ..INPUT..
         LA    R3,X'09'(,R3)      ...CHAR..
         OR    R2,R3              ....INTO DEVICE NUMBER
         LA    R1,1(,R1)          NEXT INPUT CHAR
         BCT   R0,INPSCN          LOOP IF NOT END OF INPUT
         B     INPDON             ALL INPUT DONE
INPDON   DS    0H
         CH    R0,=H'4'           ANY INPUT?
         BNE   INPOK              YES - ACCEPT
INPERR   DS    0H
         LA    R1,=CL24'** ERROR ** BAD INPUT!  '
         LA    R0,24
         TPUT  (1),(0)            ISSUE AN ERROR MESSAGE
         B     SCANLOOP           LOOP..
INPOK    DS    0H
         STH   R2,DEVNUMX         SAVE HEX DEVICE NUMBER
         LA    R3,DEVNUMX         POINT AT DEVICE NUMBER
*-------------------------------------------------------------------*
*   LOOP THROUGH ALL DASD UCBS LOOKING FOR THE DEVICE NUMBER.       *
*-------------------------------------------------------------------*
         UCBSCAN COPY,                                              X
               WORKAREA=SCANWORK,                                   X
               UCBAREA=UCBSTOR,                                     X
               RANGE=ALL,                                           X
               DEVCLASS=ALL,                                        X
               DEVN=(R3),                                           X
               MF=(E,SCANLIST)
*-------------------------------------------------------------------*
*   IF UCBSCAN RETURNED A UCB, CHECK WHETHER THE UCB REPRESENTS     *
*   THE SYSRES VOLUME.  IF IT ISN'T, CONTINUE CHECKING MORE UCBS.  IF *
*   THE UCB REPRESENTS THE SYSRES DEVICE, END THE LOOP.             *
*-------------------------------------------------------------------*
         STM   R15,R0,ERRORQ      SAVE RC / RSN
         LA    R1,=CL24'** ERROR ** UCBSCAN     '
         LA    R0,24
         CLI   ERRORQ+03,X'04'    RC = 4
         BE    UCBNFNT            YES - CHECK FOR NOT FOUND
         LTR   R15,R15            TEST RETURN CODE
         BNZ   EREXITS            EXIT IF NON-ZERO
         CLC   UCBCHAN,DEVNUMX    RIGHT UCB?
         BE    PATHINF
         MVC   PTHINF1(24),=CL24'** NEXT UCB XXXX'
         UNPK  PTHINF1+12(5),UCBCHAN(3)
         TR    PTHINF1+12(4),H2C-C'0'
```

```
         MVI   PTHINF1+16,C' '
         LA    R1,PTHINF1
         LA    R0,24
         TPUT  (1),(0)
UCBNFND  DS    0H
         LA    R1,=CL24'** ERROR ** NO UCB FOUND'
         LA    R0,24
         TPUT  (1),(0)
         B     SCANLOOP
UCBNFNT  DS    0H
         CLI   ERRORQ+07,X'01'    RSN = 1
         BE    UCBNFND            YES - TELL 'NOT FOUND'
         B     EREXITS            EXIT IF NON-ZERO
*.....................................................................*
*   ISSUE THE UCBINFO MACRO TO OBTAIN PATH-RELATED INFORMATION.       *
*   UCBINFO RETURNS THIS INFORMATION IN A FIELD CALLED PATHSTOR,      *
*   MAPPED BY IOSDPATH.                                               *
*.....................................................................*
PATHINF  DS    0H
         UCBINFO PATHINFO,                                        X
               DEVN=UCBCHAN,                                      X
               PATHAREA=PATHSTOR,                                 X
               MF=(E,INFOLIST)
*.....................................................................*
*   IF UCBINFO CANNOT RETRIEVE PATH-RELATED INFORMATION, THAT IS, YOU *
*   RECEIVE A NON-ZERO RETURN CODE, EXIT PROGRAM.                     *
*.....................................................................*
         STM   R15,R0,ERRORQ      SAVE RC / RSN
         LA    R1,=CL24'** ERROR ** UCBINFO PATH'
         LA    R0,24
         LTR   R15,R15           TEST FOR 0 RETURN CODE
         BNZ   EREXITI           EXIT IF BAD RC
*.....................................................................*
*   LOOP THROUGH THE CHANNEL PATH ID ARRAY ENTRIES RETURNED IN        *
*   PATHSTOR AND BUILD A STATUS INFORMATION FOR EACH PATH.            *
*.....................................................................*
         MVC     PTHINF1,=CL48' '  CLEAR..
         MVC     PTHINF2,=CL48' '  .STATUS..
         MVC     PTHINF3,=CL48' '  ..MESSAGE..
         MVC     PTHINF4,=CL48' '  ...DATA
         MVC     PTHINF1+00(6),=CL6'DEVICE'
         UNPK    PTHINF1+07(5),UCBCHAN(3)
         TR      PTHINF1+07(4),H2C-C'0'
         MVI     PTHINF1+11,C' '
         MVC     PTHINF1+12(8),=CL8'OFFLINE'
         TM      UCBSTAT,UCBONLI   ONLINE?
         BNO     *+4+6
         MVC     PTHINF1+12(8),=CL8'ONLINE'
         TM      UCBJBNR,UCBJES3   JES3 MANAGED?
         BNO     *+4+6
         MVC     PTHINF1+20(6),=CL6'(JES3)'
         LA      R10,PATHSTOR      ADDRESS OF PATHINFO DATA
         USING   PATH,R10          SET UP ADDRESSABILITY TO
*                                  PATH INFORMATION.
         MVC     PTHINF1+27(18),=CL18'PATH VALIDATED'
         TM      PATHFLAGS,PATHVALPH PATH VALIDATED?
         BZ      *+4+6
         MVC     PTHINF1+27(18),=CL18'PATH NOT VALIDATED'
         ICM     R2,15,PATH#CHPIDS NBR OF CHPIDS.
         BNZ     CHPID_OK
         LA      R1,PTHINF1
         LA      R0,L'PTHINF1
         TPUT    (1),(0)
         LA      R1,=CL24'** ERROR ** NO PATH INFO'
         LA      R0,24
         TPUT    (1),(0)
         B     SCANLOOP
CHPID_OK DS      0H
         SR      R3,R3             CHPID SAVE ARRAY INDEX REGISTER.
         SR      R8,R8             CHPID ARRAY INDEX REGISTER.
CHPID_LO DS      0H
         IC      R11,PATHBITS(R8)  GET FLAGS FROM ARRAY ENTRY.
         STC     R11,PATHSAVE(R3)  SAVE ENTRY
         LH      R11,PATHCHPID(R8) GET THE ID FOR THE CHPID
         STC     R11,CHPID(R3)     SAVE THE ID
         LA      R8,L'PATHCHPIDARRAY(,R8)   INCREMENT ARRAY INDEX
         LA      R3,1(,R3)
         BCT     R2,CHPID_LO
*.....................................................................*
*        CREATE PATH STAUS INFO.                                      *
*.....................................................................*
         MVC     PTHINF2(18),=CL18'CHP             '
         MVC     PTHINF3(18),=CL18'CHP PHYS AVAILABE'
         MVC     PTHINF4(18),=CL18'CHP LOGI AVAILABE'
         L       R2,PATH#CHPIDS NBR OF CHPIDS.
         LA      R10,PTHINF2+18
         LA      R14,PTHINF3+18
         LA      R15,PTHINF4+18
         LA      R3,CHPID
         LA      R4,PATHSAVE
         SR      R8,R8
CHPID_EX DS      0H
         UNPK    0(3,R10),0(2,R3)
         TR      0(2,R10),H2C-C'0'
         MVI     2(R10),C' '
         MVI     0(R14),C'N'       PHYS AVAILABLE
         TM      0(R4),PATHPAM
         BZ      *+4+4
         MVI     0(R14),C'Y'       PHYS AVAILABLE
         MVI     0(R15),C'N'       LOG AVAILABLE
         TM      0(R4),PATHLPM
         BZ      *+4+4
         MVI     0(R15),C'Y'       LOG AVAILABLE
*.....................................................................*
         LA      R10,3(,R10)
         LA      R14,3(,R14)
         LA      R15,3(,R15)
         LA      R3,1(,R3)
         LA      R4,1(,R4)
         BCT     R2,CHPID_EX       LOOP.....
*.....................................................................*
*    DISPLAY STATUS                                                   *
*.....................................................................*
         LA      R2,(PTHINFE-PTHINF1)/L'PTHINF1
         LA      R3,PTHINF1
CHPID_TP DS      0H
         LA      R0,L'PTHINF1
         TPUT    (3),(0)
         LA      R3,L'PTHINF1(,R3)
         BCT     R2,CHPID_TP       LOOP.....
*.....................................................................*
*   TEST THE AVAILABILITY OF THE PATHS TO THE DEVICE THROUGH          *
*   ALL CHPIDS USING THE IOSPTHV MACRO.  SUPPLY THE CHANNEL           *
*   PATH ID OF THE ONLINE PATH ON THE CHPID PARAMETER.                *
*                                                                     *
*   NOTE: ALTHOUGH THE LOGICAL PATH MASK (LPM) INDICATED THAT         *
*         THE PATH WAS LOGICALLY ONLINE TO THE DEVICE, IT IS          *
*         POSSIBLE THAT THE PATH IS NOT OPERATIONAL.  IOSPTHV         *
*         PERFORMS AN I/O OPERATION DOWN THE PATH TO                  *
*         DETERMINE IF A NON-OPERATIONAL CONDITION EXITS.             *
*.....................................................................*
         LA      R10,PATHSTOR
         L       R2,PATH#CHPIDS NBR OF CHPIDS.
         LA      R3,CHPID
CHPID_PV DS      0H
         IOSPTHV DEVN=UCBCHAN,                                    X
               CHPID=(R3),                                        X
               MF=(E,PTHVLIST)
*.....................................................................*
*   A ZERO RETURN CODE INDICATES AN OPERATIONAL PATH TO               *
*   THE SPECIFIED DEVICE.  A NON-ZERO RETURN CODE INDICATES           *
*   A NON-OPERATIONAL PATH.  IN THE LATTER CASE, EXAMINE THE          *
*   RETURN AND REASON CODE TO DETERMINE THE CAUSE.                    *
*.....................................................................*
         STM   R15,R0,ERRORQ
         MVC   PTHINF1,=CL48' '  CLEAR..
         MVC   PTHINF2,=CL48' '  CLEAR..
         MVC   PTHINF1+00(6),=CL6'DEVICE'
         UNPK  PTHINF1+07(5),UCBCHAN(3)
         TR    PTHINF1+07(4),H2C-C'0'
         MVI   PTHINF1+11,C' '
         MVC   PTHINF1+12(6),=CL6'CHPID'
         UNPK  PTHINF1+18(3),0(2,R3)
         TR    PTHINF1+18(2),H2C-C'0'
         MVI   PTHINF1+20,C' '
         MVC   PTHINF1+21(18),=CL18'VALIDATION RC/RSN'
         UNPK  PTHINF1+39(9),ERRORQ+00(5) RC
         TR    PTHINF1+39(8),H2C-C'0'
         UNPK  PTHINF1+48(9),ERRORQ+04(5) RSN
         TR    PTHINF1+48(8),H2C-C'0'
         MVI   PTHINF1+47,C'/'
         MVI   PTHINF1+56,C' '
         MVC   PTHINF1+57(4),=CL4'O.K.'
         LTR   R15,R15
         BZ    PATH_OK
PATH_NOK DS    0H
         MVC   PTHINF1+56(21),=CL21' PROGRAM/SYSTEM ERROR'
         CH    R15,=H'4'
         BH    PATH_OK
         MVC   PTHINF1+56(21),=CL21' PATH NOT AVAILABLE  '
         CH    R0,=H'4'
         BE    PATH_OK
         MVC   PTHINF1+56(21),=CL21' TIME_OUT            '
PATH_OK  DS    0H
         LA    R1,PTHINF1
         LA    R0,79
         TPUT  (1),(0)
         LA    R3,1(,R3)
         BCT   R2,CHPID_PV
         B     SCANLOOP
*.....................................................................*
*    RETURN A MESSAGE TO TELL THE USER THAT THE                       *
*    IOSPTHV MACRO IS NOT AVAILABLE ON THE SYSTEM EXECUTING           *
*    THIS SAMPLE PROGRAM.                                             *
*.....................................................................*
NOSUPP   DS    0H
         LA    R1,=CL24'** ERROR ** IOSPTHV N/A '
         LA    R0,24
         TPUT  (1),(0)
         B     EXIT
*.....................................................................*
*    ISSUE ERROR MESSAGE                                              *
*.....................................................................*
EREXITS  DS    0H
         TPUT  (1),(0)
         UNPK  SCANWORK+00(9),ERRORQ+00(5) RC
```

```
         TR    SCANWORK+00(8),H2C-C'0'
         UNPK  SCANWORK+09(9),ERRORQ+04(5) RSN
         TR    SCANWORK+09(8),H2C-C'0'
         MVC   SCANWORK+17(21),=CL21' PROGRAM/SYSTEM ERROR'
         LA    R1,SCANWORK
         LA    R0,38
         TPUT  (1),(0)
         B     SCANLOOP
EREXITI  DS    0H
         TPUT  (1),(0)
         UNPK  SCANWORK+00(9),ERRORQ+00(5) RC
         TR    SCANWORK+00(8),H2C-C'0'
         UNPK  SCANWORK+09(9),ERRORQ+04(5) RSN
         TR    SCANWORK+09(8),H2C-C'0'
         MVC   SCANWORK+17(21),=CL21' PROGRAM/SYSTEM ERROR'
         CLI   ERRORQ+03,X'18'
         BNE   EREXITIP
         MVC   SCANWORK+17(21),=CL21' SUBCHANNEL ERROR    '
         CLI   ERRORQ+07,X'08'
         BNE   EREXITIP
         MVC   SCANWORK+17(21),=CL21' UCB NOT CONNECTED   '
EREXITIP DS    0H
         LA    R0,38
         LA    R1,SCANWORK
         TPUT  (1),(0)
         B     SCANLOOP
*.....................................................................*
*    EXIT FINAL......                                                 *
*.....................................................................*
EXIT     DS    0H
         L     R13,4(,R13)             RELOADS CALLER'S SAVE
*                                              AREA ADDR INTO 13
         LA    R0,DYNSIZE              LOADS DYNAMIC AREA SIZE
         STORAGE RELEASE,SP=233,ADDR=(R6),LENGTH=(R0)
         MODESET KEY=NZERO,MODE=PROB
         BAL   R14,SETPM
         RETURN (14,12),RC=0           RETURN
*.....................................................................*
*    AUTHORIZE                                                        *
*.....................................................................*
SETSM    DS    0H
         L     2,X'21C'(,0)       TCB ADDR.
         USING TCB,2
         L     15,TCBJSCB         JSCB ADDR.
         LA    1,DYTHS            INVOKE..
         SVC   X'DC'              .USER SVC..
         BR    14                 ..TO SET..
DYTHS    OI    X'EC'(15),X'01'    ...APF AUTH ON TEMPORARILY.
SETPM    DS    0H
         L     2,X'21C'(,0)       TCB ADDR.
         USING TCB,2
         L     15,TCBJSCB         JSCB ADDR.
         LA    1,DYTHSR           INVOKE..
         SVC   X'DC'              .USER SVC..
         BR    14                 ..TO SET..
DYTHSR   NI    X'EC'(15),X'FF'-X'01' APF AUTH OFF PERMANETLY.
*.....................................................................*
*    DSECTS TO MAP SAVE AREAS AND DYNAMIC AREA                        *
*.....................................................................*
H2C      DC    C'0123456789ABCDEF'
*
         LTORG
*.....................................................................*
DYNAREA  DSECT
SAVE     DS    18F            SAVE AREA
         DS    0D             FORCE DOUBLEWORD ALIGNMENT
DEVNUMC  DS    CL4            DEVICE NUMBER CHAR
DEVNUMX  DS    H              DEVICE NUMBER HEX
ERRORQ   DS    2F             ERROR QUALIFIER
*.....................................................................*
*  LIST FORMS OF MACROS.  THE LIST AND EXECUTE FORMS OF THESE MACROS  *
*  ARE USED BECAUSE THIS MODULE IS REENTRANT.                         *
*.....................................................................*
LIST_INFOSERV UCBINFO MF=(L,INFOLIST)   LIST FORM OF UCBINFO
INFOSERV_END  DS   0D
PATHSTOR      DS   CL256                STORAGE FOR THE PATHAREA
PATHSTOR_END  DS   0D
LIST_PTHVSERV IOSCDR  MF=(L,PTHVLIST)   LIST FORM OF IOSPTHV
PTHVSERV_END  DS   0D
LIST_SCANSERV UCBSCAN MF=(L,SCANLIST)   LIST FORM OF UCBSCAN
SCANSERV_END  DS 0D
SCANWORK      DS CL100                  SCAN WORK AREA
SCANWORK_END  DS 0D
UCBSTOR       DS CL48                   UCB COPY STORAGE
UCBSTOR_END   DS 0D
*.....................................................................*
PTHINF1       DS CL48
PTHINF2       DS CL48
PTHINF3       DS CL48
PTHINF4       DS CL48
PTHINFE       DS 0C
*.....................................................................*
CHPID         DS XL8                CHPID USED FOR IOSCDR INVOCATION
PATHSAVE      DS XL8                WORK VARIABLE FOR CHPID ARRAY
*                                   ENTRIES IN THE PATHAREA.
END_DYN       DS    0D
DYNSIZE       EQU   *-DYNAREA       CALCULATES DYNAMIC AREA
*.....................................................................*
         PUSH PRINT
         PRINT NOGEN
         CVT  LIST=YES,DSECT=YES
         POP  PRINT
         IOSDPATH
         TITLE ' '
UCB      DSECT
         IEFUCBOB
         IKJTCB
         END   DEVACC
```

Appendix B. Sample Program for Subspace Management

This following example is for illustration purposes only. Please refer to *MVS/ESA SP V5 Auth Assembler Services Reference ENF-ITT* for detailed usage notes.

```
*     OBTAIN THE STORAGE FROM A PAGEABLE SUBPOOL
         STORAGE OBTAIN,LENGTH=4096*(256+256),BNDRY=PAGE,SP=0,COND=YES
         ST    1,STORSTRT
         LTR   15,15                      IF NOT SUCCESSFUL (0)
         BNZ   NOSTOR                     GO TO ERROR PROCESSING
*
*     MAKE IT SEGMENT ALIGNED
*
         L     9,ROUNDIT
         L     2,ONEMEG
         L     10,STORSTRT
         ALR   10,2
         NR    10,9
         ST    10,STORSEGA              NEW SEGMENT-ALIGNED BOUNDARY
         L     1,STORSEGA
         ST    1,RPTR1                  PUT IT IN THE RANGE LIST
*  ******************************************************************
*     CREATE 5 SUBSPACES
*  ******************************************************************
         LA    5,1                        INIT LOOP COUNTER
         LA    9,STOKEN1                  START WITH FIRST STOKEN
*                                         IN ARRAY
LOOP1    DS    OH
         IARSUBSP CREATE,NAME=SSNAME,STOKEN=(9),                       *
               GENNAME=COND,OUTNAME=ONAME
         LTR   15,15                      IF NOT SUCCESSFUL (0)
         BNZ   NOCREATE                   GO TO ERROR PROCESSING
         LA    4,1                        LOOP INCREMENT IS 1
         ALR   5,4                        BUMP UP LOOP COUNTER
         LA    10,8                       ARRAY INCREMENT IS 8
         ALR   9,10                       BUMP UP ARRAY INDEX
         LA    4,5
         CR    5,4                        CHECK HOW MANY SO FAR
         BNH   LOOP1                      IF NOT 5 YET, REPEAT
*  ******************************************************************
*  ******************************************************************
*     ADD THE SUBSPACE ENTRY TO THE WORKUNIT ACCESS LIST
*  ******************************************************************
         ALESERV ADD,STOKEN=STOKEN1,ALET=SSALET,AL=WORKUNIT
*  ******************************************************************
*     MAKE THE STORAGE SUBSPACE-ELIGIBLE
*  ******************************************************************
         IARSUBSP IDENTIFY,RANGLIST=RANGPTR,NUMRANGE=NUMRANG
*  ******************************************************************
*     ASSIGN THE STORAGE TO THE SUBSPACE
*  ******************************************************************
         IARSUBSP ASSIGN,STOKEN=STOKEN1,RANGLIST=RANGPTR
*  ******************************************************************
*     BRANCH TO THE SUBSPACE
*  ******************************************************************
          L     2,=A(X'80000000'+NEXT1)
          BSG   0,2
*  ******************************************************************
*     RUN PROGRAM IN THE SUBSPACE
*  ******************************************************************
*    RETURN TO THE BASE SPACE (FULL ADDRESS SPACE ADDRESSABILITY)
*  ******************************************************************
NEXT1     DS    OH
          L     0,=A(X'80000000'+NEXT2)
          BSG   0,0
*  ******************************************************************
*  DISASSOCIATE THE STORAGE (NUMRANGE DEFAULTS TO 1 WHICH IS WHAT WE HAVE)
*  ******************************************************************
NEXT2     DS    OH
         IARSUBSP UNASSIGN,STOKEN=STOKEN1,RANGLIST=RANGPTR
*  ******************************************************************
*  MAKE THE STORAGE INELIGIBLE TO BE ASSIGNED TO A SUBSPACE
*  ******************************************************************
         IARSUBSP UNIDENTIFY,RANGLIST=RANGPTR
*  ******************************************************************
*  DELETE THE SUBSPACE
*  ******************************************************************
*
         IARSUBSP DELETE,STOKEN=STOKEN1
*
*  ******************************************************************
*  SUBSPACE CREATE FAILED - RELEASE THE STORAGE
*  ******************************************************************
NOCREATE DS    OH                     ERROR EXIT POINT
*  ******************************************************************
*  RELEASE THE STORAGE - USE THE ORIGINAL ADDRESS STORSTRT
*  ******************************************************************
         STORAGE RELEASE,ADDR=STORSTRT,LENGTH=4096*(256+256)
*  ******************************************************************
*  STORAGE OBTAIN FAILED - UNDO WHATEVER STEPS HAD BEEN SUCCESSFUL
*  PRIOR TO THE STORAGE OBTAIN
*  ******************************************************************
NOSTOR   DS    OH                     Error exit point
          .
          .
          .
          .
          .
          .
*  ******************************************************************
*  DECLARES
*  ******************************************************************
ONEMEG   DC    F'1048576'             ONE MEGABYTE
ROUNDIT  DC    X'FFF00000'            ROUND IT TO A SEGMENT ADDRESS
SSNAME   DC    CL8'SSPACE1 '          SUBSPACE NAME
ONAME    DS    CL8                    GENERATED NAME IF NEEDED
SSSTOKEN DS    OCL40
STOKEN1  DS    CL8
STOKEN2  DS    CL8
STOKEN3  DS    CL8
STOKEN4  DS    CL8
STOKEN5  DS    CL8
SSALET   DS    5CL4
STORSTRT DS    1F                     ADDRESS FOR OBTAIN/RELEASE
STORSEGA DS    1F                     SEGMENT-ALIGNED ADDRESS
*
* RANGE LIST MAPPING
*
RLIST    DS    OCL8
RPTR1    DS    F
NUMBLKS  DC    F'256'
*
RANGPTR  DC    A(RLIST)
NUMRANG  DC    F'1'
```

Appendix C. Coding Sample for Enclave Implementation

The following assembly source outlines a basic application using enclaves. The parmeters shown are minimal and all the macros should get RC=0 if run as is, either as a job or as an STC.

```
************************************************************
**   Must be link-edited with AMODE=31 and AC=1           **
**   Must reside in an authorized library                 **
************************************************************
ENCLTEST CSECT
         SAVE  (14,12)
         BALR  12,0
         USING *,12
         ST    13,SAPTR
         B     GO
         DC    CL8'ENCLTEST'
SAPTR    DS    1F
CLSFYSA  DS    18F
GENSUB   DC    CL4'STC '
SUBNAME  DC    CL8'ENCL    '
CONNTOK  DS    FL4
CLSFYTOK DS    FL4
ETOK     DS    FL4
ARRVLT   DS    2F
EFUNCT   DC    CL8'ANY_NAME'
         IWMCLSFY MF=(L,CLSFYPL)
************************************************************
**   Must run in supervisor or pkm 0-7                    **
************************************************************
GO       MODESET  KEY=ZERO
************************************************************
**   Connection to WLM is required                        **
************************************************************
         IWMCONN  SUBSYS=GENSUB,SUBSYSNM=SUBNAME,
               CONNTKN=CONNTOK,CONNTKNKEYP=PSWKEY
         XR    2,2
         CR    2,15
         BE    CONNRCO
         WTO   'IWMCONN  RC NOT 0'
         B     LEAVE
CONNRCO  WTO   'IWMCONN  RC 0'
************************************************************
**   Peek incoming service requests                       **
************************************************************
*        SOME INTERFACE CODE HERE.
*        (ENCLAVES SHOULD NOT BE INACTIVE FOR MORE THAN A
*         FEW SECONDS. DELETE THEM IF NO NEW WORK ARRIVES)
         STCK  ARRVLT
************************************************************
**   Assign service clasiffy parameters                   **
************************************************************
         LA    13,CLSFYSA
         IWMCLSFY CONNTKN=CONNTOK,MF=(M,CLSFYPL)
*        ...
         XR    2,2
         CR    2,15
         BE    CLSFYRCO
         WTO   'IWMCLSFY RC NOT 0'
         B     DISC
CLSFYRCO WTO   'IWMCLSFY RC 0'
************************************************************
**   Create an enclave   (or skip if current is valid)    **
************************************************************
         IWMECREA CLSFY=CLSFYPL,ETOKEN=ETOK,
               ARRIVALTIME=ARRVLT,FUNCTION_NAME=EFUNCT
*        ...
         XR    2,2
         CR    2,15
         BE    ECREARCO
         WTO   'IWMECREA RC NOT 0'
         B     DISC
ECREARCO WTO   'IWMECREA RC 0'
************************************************************
**   Schedule SRB                                         **
************************************************************
         MODESET MODE=SUP
         IEAMSCHD EPADDR=MYSRB,PRIORITY=ENCLAVE,
               ENCLAVETOKEN=ETOK
*        ...
         XR    2,2
         CR    2,15
         BE    IEAMSRCO
         WTO   'IEAMSCHD RC NOT 0'
         B     PRBLMOD
IEAMSRCO WTO   'IEAMSCHD RC 0'
PRBLMOD  MODESET MODE=PROB
*        GO BACK TO PEEK NEXT REQUEST
************************************************************
**   Delete the enclave  (when finished)                  **
************************************************************
         IWMEDELE ETOKEN=ETOK
         XR    2,2
         CR    2,15
         BE    EDELERCO
         WTO   'IWMEDELE RC NOT 0'
         B     DISC
EDELERCO WTO   'IWMEDELE RC 0'
************************************************************
**   Disconnect from WLM                                  **
************************************************************
DISC     IWMDISC  CONNTKN=CONNTOK
************************************************************
**   Exit                                                 **
************************************************************
LEAVE    L     13,SAPTR
         RETURN (14,12),,RC=(15)
************************************************************
**   SRB Routine                                          **
************************************************************
MYSRB    NOP   0
*        ...
         BR    14
************************************************************
**   Required mappings for MODESET macro                  **
************************************************************
         IKJTCB
         IHARB
************************************************************
**   Required mappings for IWMxxxxx macros                **
************************************************************
         IWMYCON
         CVT   PREFIX=NO,DSECT=YES
************************************************************
**   That's all                                           **
************************************************************
         END
```

Appendix D. Coding Sample for Shared Pages Implementation

The following assembly source outlines basic applications exploiting shared pages. It also contains dumps of shared storage to illustrate the results of sharing.

D.1 Sample Program Sharing Data Above and Below 16MB

The following sample program gets storage above the 16MB and below 16MB. The buffer above 16MB is updated. The rangelist mapped by IARVRL is initialized and the IARVSERV macro executed with TARGET_VIEW=SHAREDWRITE. Both virtual buffers now share the same real page. The target buffer below 16MB has the same data in it as the source buffer above 16MB. The program cleans up by issuing the IARVSERV UNSHARE macro, storage release and return.

```
CONTROL  TITLE 'SHARED PAGES'
*****************************************************************
*                                                               *
*    IARVSERV SHARE TARGET_VIEW=SHAREDWRITE                     *
*                                                               *
*****************************************************************
         SPACE
CONTROL  AMODE 31                    31-BIT ADDRESSING MODE.
CONTROL  RMODE ANY                   MAY RESIDE ABOVE OR BELOW 16MB.
         SPACE 2
CONTROL  START 0                     DEFINE CSECT NAME AND INITIALIZE
*                                    A LOCATION COUNTER TO ADDRESS 00.
         SPACE
         GBLC  &SYSSPLV              DEFINE THE GLOBAL VARIABLE THAT
*                                    WILL BE SET BY THE SPLEVEL MACRO.
         SPACE
         SPLEVEL TEST                THIS MACRO SETS &SYSSPLV TO:
*                                       1 > ASSEMBLING O/S IS MVS/370.
*                                       2 > ASSEMBLING O/S IS MVS/XA.
*                                       3 > ASSEMBLING O/S IS MVS/ESA.
         TITLE 'STANDARD REGISTER EQUATES'
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         TITLE 'SHARED PAGES CODING AND VIRTUAL STORAGE INTERFACES'
SAVEREGS EQU   *
         SPACE
         BAKR  R14,0                 SAVE CALLER'S ARS, GPRS, AND    .
*                                    RETURN ADDRESS ON LINKAGE STACK.
IDENTIFY EQU   *
         SPACE
         USING CONTROL,R15           ESTABLISH TEMPORARY ADDRESABILITY.
         B     SETBEG                BRANCH AROUND IDENTIFIER.
         SPACE
         DC    CL8'CONTROL'          THE NAME OF THIS CSECT.
         DC    CL8'&SYSDATE'         ASSEMBLY DATE.
         DC    CL7'&SYSTIME'         ASSEMBLY TIME.
         DC    CL1'&SYSSPLV'         ASSEMBLING OPERATING SYSTEM.
         SPACE
*****************************************************************
*                                                               *
*    IARVSERV SHARE TARGET_VIEW=SHAREDWRITE                     *
*                                                               *
*****************************************************************
         SPACE
         DROP  R15
*****************************************************************
*         Testcase Mainline Starts Here                         *
*****************************************************************
SETBEG   EQU   *
         LAE   R12,0(R15,0)
         USING CONTROL,R12
*****************************************************************
*  1.     Assembler code starts here                            *
*****************************************************************
       SLR   2,2
       ST    2,TCRCODE              initialize testcase retcode
       XC    VRLMAP(28),VRLMAP      zero the vro
*****************************************************************
*  2. Getmain 2 areas for data                                  *
*****************************************************************
STOR1    STORAGE OBTAIN,LENGTH=4096*100,BNDRY=PAGE,SP=0
         ST    1,SRCSTART
STOR2    STORAGE OBTAIN,LENGTH=4096*100,BNDRY=PAGE,SP=0,LOC=BELOW
         ST    1,TGTSTART
*****************************************************************
*  3. Put data into 100 pages of source area                    *
*****************************************************************
         L     3,SRCSTART
         LA    10,1                   loop control is reg 10
INITLOOP DS    0H
         MVC   0(8,3),STUFF
         AL    3,PAGELEN             increment by 4096 bytes
         LA    11,1
         ALR   10,11                  update loop control
         L     4,DSPSIZE              have 100 pages to do
         CR    10,4                   if not done yet
         BNH   INITLOOP               repeat the loop
*****************************************************************
*  4. Set up rangelist for the share                            *
*****************************************************************
* source_vsa=
         LA 7,VRLMAP
         L  6,SRCSTART
         ST 6,0(7)
* source_aletstkn
         L  6,SRCALET
         ST 6,8(7)
* numpages
         L  6,DSPSIZE
         ST 6,12(7)
* target_vsa
         L  6,TGTSTART
         ST 6,16(7)
* target_aletstkn
         L  6,TGTALET
         ST 6,24(7)
* vrlptr
         LA 5,VRLMAP
         ST 5,VRLADDR
         LA 5,VRLADDR
*****************************************************************
*  5. Issue the  IARVSERV SHARE for SHAREDWRITE                 *
*****************************************************************
VSERV1   IARVSERV SHARE,RANGLIST=(5),TARGET_VIEW=SHAREDWRITE
*****************************************************************
* 5a. Test return code in R15 and reason code if not 0 in R0    *
*****************************************************************
*****************************************************************
*  6. You are now sharing data                                  *
```

```
*********************************************************************
*
*
*
*********************************************************************
*  7. Clean up stuff.                                               *
*********************************************************************
VSERV3   IARVSERV UNSHARE,RANGLIST=(5)
         L  1,SRCSTART
RELST1   STORAGE  RELEASE,SP=0,LENGTH=4096*100,ADDR=(R1)
         L  1,TGTSTART
RELST2   STORAGE  RELEASE,SP=0,LENGTH=4096*100,ADDR=(R1)
         SAC   0
         SR    R15,R15              SET COMPLETION CODE.
         PR                         RESTORES REGISTERS 2-14, AMODE,
*                                   ASCENV, AND RETURNS TO CALLER.
*********************************************************************
*            Add assembler declares here                            *
*********************************************************************
VRLPTR   DS    F                    basing pointer for VRL map
SRCSTART DS    F                    data space start address
TGTSTART DS    F                    data space start address
RANGEADR DS    F                    pointer to rangelist
VRLADDR  DS    F                    pointer to vrlptr
TCRCODE  DS    F
*
*       CONSTANTS
*
SRCALET  DC    F'0'                 data space alet
TGTALET  DC    F'0'                 data space alet
PAGELEN  DC    F'4096'              length of a page
DSPSIZE  DC    F'100'               number of pages in data space
STUFF    DC    C'INITDATA'          initialization data
*********************************************************************
*                                                                   *
* All generated literals (e.g. =C'YES' or =X'FF' or =F'-1') will be  *
* inserted after the following "LTORG" (LITERAL ORIGIN) statement.   *
*                                                                   *
*********************************************************************
         SPACE
         LTORG
*********************************************************************
*            Include mapping of the VRL                             *
*********************************************************************
VRLMAP   DS    8F                   This creates a dsect
         IARVRL
CONTROL  CSECT
         SPACE 2
         END   ,                       DENOTES END OF MY PROGRAM.
```

The target buffer starting at address 00006000, is sharing the same data as shown in Figure 81

```
LIST 00006000 ASID(X'00FB') LENGTH(102400) AREA
00006000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00006010 LENGTH(4080)==>All bytes contain X'00'
00007000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00007010 LENGTH(4080)==>All bytes contain X'00'
00008000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00008010 LENGTH(4080)==>All bytes contain X'00'
00009000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00009010 LENGTH(4080)==>All bytes contain X'00'
0000A000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0000A010 LENGTH(4080)==>All bytes contain X'00'
0000B000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0000B010 LENGTH(4080)==>All bytes contain X'00'
0000C000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0000C010 LENGTH(4080)==>All bytes contain X'00'
0000D000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0000D010 LENGTH(4080)==>All bytes contain X'00'
0000E000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0000E010 LENGTH(4080)==>All bytes contain X'00'
0000F000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0000F010 LENGTH(4080)==>All bytes contain X'00'
00010000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00010010 LENGTH(4080)==>All bytes contain X'00'
00011000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00011010 LENGTH(4080)==>All bytes contain X'00'
00012000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00012010 LENGTH(4080)==>All bytes contain X'00'
00013000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00013010 LENGTH(4080)==>All bytes contain X'00'
00014000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00014010 LENGTH(4080)==>All bytes contain X'00'
00015000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
00015010 LENGTH(4080)==>All bytes contain X'00'
00016000 LENGTH(36864)==>Storage not available
```

Figure 81. Shared Data Target Buffer Below 16MB

A dump of storage was taken after the *IAVSERV SHARE* was issued in the program. Refer to D.3.1, “RSMDATA SHRDATA Report” on page 206 for the RSM shared data report; notice that real frame *0CC5A* maps the two virtual storage adresses *PAGE 00006000 and sharing page 05701000*. The source buffer starting at *05701000* is shown in Figure 82.

```
LIST 05701000 ASID(X'00FB') LENGTH(102400) AREA
05701000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05701010 LENGTH(4080)==>All bytes contain X'00'
05702000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05702010 LENGTH(4080)==>All bytes contain X'00'
05703000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05703010 LENGTH(4080)==>All bytes contain X'00'
05704000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05704010 LENGTH(4080)==>All bytes contain X'00'
05705000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05705010 LENGTH(4080)==>All bytes contain X'00'
05706000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05706010 LENGTH(4080)==>All bytes contain X'00'
05707000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05707010 LENGTH(4080)==>All bytes contain X'00'
05708000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05708010 LENGTH(4080)==>All bytes contain X'00'
05709000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05709010 LENGTH(4080)==>All bytes contain X'00'
0570A000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0570A010 LENGTH(4080)==>All bytes contain X'00'
0570B000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0570B010 LENGTH(4080)==>All bytes contain X'00'
0570C000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0570C010 LENGTH(4080)==>All bytes contain X'00'
0570D000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0570D010 LENGTH(4080)==>All bytes contain X'00'
0570E000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0570E010 LENGTH(4080)==>All bytes contain X'00'
0570F000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
0570F010 LENGTH(4080)==>All bytes contain X'00'
05710000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05710010 LENGTH(4080)==>All bytes contain X'00'
05711000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05711010 LENGTH(4080)==>All bytes contain X'00'
05712000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05712010 LENGTH(4080)==>All bytes contain X'00'
05713000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05713010 LENGTH(4080)==>All bytes contain X'00'
05714000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05714010 LENGTH(4080)==>All bytes contain X'00'
05715000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05715010 LENGTH(4080)==>All bytes contain X'00'
05716000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05716010 LENGTH(4080)==>All bytes contain X'00'
05717000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05717010 LENGTH(4080)==>All bytes contain X'00'
05718000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05718010 LENGTH(4080)==>All bytes contain X'00'
05719000. C9D5C9E3 C4C1E3C1 00000000 00000000 |INITDATA........|
05719010 LENGTH(4080)==>All bytes contain X'00'
```

Figure 82. Shared Data Source Buffer Above 16MB

D.2 Sample Program Sharing Data in Data Spaces

The example sets up two data spaces and initializes the source data space with data. Then sets up sharing with the target data space by issuing the *IARVSERV* macro. We can now update or view the data in either data space, as the data is shared between both data spaces. Finally we clean up by deleting the data spaces and returning using *IARVSERV UNSHARE* service.

```
CONTROL  TITLE 'SHARED PAGES'
*********************************************************************
*                                                                   *
*    IARVSERV SHARE TARGET_VIEW=SHAREDWRITE                         *
*                                                                   *
*********************************************************************
         SPACE
CONTROL  AMODE 31                    31-BIT ADDRESSING MODE.
CONTROL  RMODE ANY                   MAY RESIDE ABOVE OR BELOW 16MB.
         SPACE 2
CONTROL  START 0                     DEFINE CSECT NAME AND INITIALIZE
*                                    A LOCATION COUNTER TO ADDRESS 00.
         SPACE
         GBLC  &SYSSPLV              DEFINE THE GLOBAL VARIABLE THAT
*                                    WILL BE SET BY THE SPLEVEL MACRO.
         SPACE
         SPLEVEL TEST                THIS MACRO SETS &SYSSPLV TO:
*                                       1 > ASSEMBLING O/S IS MVS/370.
*                                       2 > ASSEMBLING O/S IS MVS/XA.
*                                       3 > ASSEMBLING O/S IS MVS/ESA.
         TITLE 'STANDARD REGISTER EQUATES'
R0       EQU   0
R1       EQU   1
R2       EQU   2
```

```
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         TITLE 'SHARED PAGES CODING AND VIRTUAL STORAGE INTERFACES'
SAVEREGS EQU   *
         SPACE
         BAKR  R14,0                SAVE CALLER'S ARS, GPRS, AND    .
*                                   RETURN ADDRESS ON LINKAGE STACK.
IDENTIFY EQU   *
         SPACE
         USING CONTROL,R15          ESTABLISH TEMPORARY ADDRESABILITY.
         B     SETBEG               BRANCH AROUND IDENTIFIER.
         SPACE
         DC    CL8'CONTROL'         THE NAME OF THIS CSECT.
         DC    CL8'&SYSDATE'        ASSEMBLY DATE.
         DC    CL7'&SYSTIME'        ASSEMBLY TIME.
         DC    CL1'&SYSSPLV'        ASSEMBLING OPERATING SYSTEM.
         SPACE
*********************************************************************
*                                                                   *
*    IARVSERV SHARE TARGET_VIEW=SHAREDWRITE                         *
*                                                                   *
*********************************************************************
         SPACE
         DROP  R15
*****************************************************************
*          Testcase Mainline Starts Here                        *
*****************************************************************
SETBEG   EQU   *
         LAE   R12,0(R15,0)
         USING CONTROL,R12
*****************************************************************
*  1.    Assembler code starts here                             *
*****************************************************************
         SLR   2,2
         ST    2,TCRCODE              initialize testcase retcode
         XC    RANGELST(256),RANGELST zero the rangelist
         XC    VRLMAP(28),VRLMAP      zero the vro
*****************************************************************
*  2.    Create 2 data spaces                                   *
*****************************************************************
DSP1A    DSPSERV CREATE,STOKEN=DSPSTOKE,NAME=DSNAME1,                *
               BLOCKS=DSPSIZE,SCOPE=SINGLE,DREF=NO,                  *
               ORIGIN=DSPSTART
DSP1B    DSPSERV CREATE,STOKEN=DSPSTOK2,NAME=DSNAME2,                *
               BLOCKS=DSPSIZE,SCOPE=SINGLE,DREF=NO,                  *
               ORIGIN=DSPSTRT2
*****************************************************************
*  3.    do aleserv add
*****************************************************************
ALE1     ALESERV ADD,STOKEN=DSPSTOKE,ACCESS=PUBLIC,AL=PASN,          *
               ALET=DSPALET
ALE2     ALESERV ADD,STOKEN=DSPSTOK2,ACCESS=PUBLIC,AL=PASN,          *
               ALET=DSPALET2
*****************************************************************
*  4. Put data into 900 pages of first data space (source)      *
*****************************************************************
         SAC   512
         L     3,DSPSTART
         L     11,DSPALET
         SAR   3,11
         LA    10,1                  loop control is reg 10
INITLOOP DS    0H
         MVC   0(8,3),STUFF
         AL    3,PAGELEN             increment by 4096 bytes
         LA    11,1
         ALR   10,11                 update loop control
         L     4,DSPSIZE             have 900 pages to do
         CR    10,4                  if not done yet
         BNH   INITLOOP              repeat the loop
*****************************************************************
*  5. Set up rangelist for the share
*****************************************************************
* source_vsa=
         LA 7,VRLMAP
         L  6,DSPSTART
         ST 6,0(7)
* source_aletstkn
         L  6,DSPALET
         ST 6,8(7)
* numpages
         L  6,DSPSIZE
         ST 6,12(7)
* target_vsa
         L  6,DSPSTRT2
         ST 6,16(7)
* target_aletstkn
         L  6,DSPALET2
         ST 6,24(7)
* vrlptr
         LA 5,VRLMAP
         ST 5,VRLADDR
         LA 5,VRLADDR
*****************************************************************
*  6. issue the  IARVSERV SHARE for SHAREDWRITE                 *
*****************************************************************
VSE3V1   IARVSERV SHARE,RANGLIST=(5),TARGET_VIEW=SHAREDWRITE
*****************************************************************
*  7. Check return code and if not 0 check reason code          *
*****************************************************************
*
*
*
*****************************************************************
*  8. You are now sharing data                                  *
*****************************************************************
*
*
*
*****************************************************************
*  9. Clean up stuff.                                           *
*****************************************************************
VSE3V3   IARVSERV UNSHARE,RANGLIST=(5)
DSP5A    DSPSERV DELETE,STOKEN=DSPSTOKE
DSP5B    DSPSERV DELETE,STOKEN=DSPSTOK2
         SAC   0
         SR    R15,R15            set completion code
         PR                       restores registers 2-14, amode,
*                                 ascenv, and returns to caller.
*****************************************************************
*            Add assembler declares here                        *
*****************************************************************
VRLPTR   DS    F                  basing pointer for VRL map
DSPALET  DS    F                  data space alet
DSPALET2 DS    F                  data space alet
DSPSTART DS    F                  data space start address
DSPSTRT2 DS    F                  data space start address
RANGEADR DS    F                  pointer to rangelist
VRLADDR  DS    F                  pointer to vrlptr
DSPSTOKE DS    2F                 data space stoken
DSPSTOK2 DS    2F                 data space stoken
RANGELST DS    CL256              storage for max numbers of ranges
TCRCODE  DS    F
*
*        CONSTANTS
*
NUMRANGS DC    F'9'               number of ranges to process
PAGELEN  DC    F'4096'            length of a page
NUMPAGES DC    FL3'50'             number of pages in each entry
NEXTRANG DC    F'409600'          bytes in 100 pages
DSPSIZE  DC    F'900'             number of pages in data space
DSNAME1  DC    C'DS1RNG06'        data space name
DSNAME2  DC    C'DS2RNG06'        data space name
STUFF    DC    C'INITDATA'        initialization data
*********************************************************************
*                                                                   *
* All generated literals (e.g. =C'YES' or =X'FF' or =F'-1') will be  *
* inserted after the following "LTORG" (LITERAL ORIGIN) statement.   *
*                                                                   *
*********************************************************************
         SPACE
         LTORG
*****************************************************************
*            Include mapping of the VRL                         *
*****************************************************************
VRLMAP   DS    8F                 This creates a dsect
         IARVRL
CONTROL  CSECT
         SPACE 2
         END   ,                     DENOTES END OF MY PROGRAM.
```

D.2.1 Sample Program Coding of IARR2V

The IARR2V macro provides a simple method to obtain a virtual storage address from a central storage address. This can be useful when you are working with an I/O or diagnostic program with a central storage address and want to get the virtual storage address.

The details of the syntax and parameters of IARR2V are contained in *MVS/ESA SP V5 Auth Assembler Services Reference ENF-ITT*. In its simplest form, the IARR2V macro only requires the RSA parameter input. The RSA parameter specifies the central storage address that you want to convert.

The VSA parameter returns the virtual storage address. The ASID, and STOKEN parameters will return the ASID and STOKEN associated with the address or data space.

If you require knowledge of whether the central storage address you have is being shared using the IARVSERV macro services, you can get that information using the WORKREG, NUMVIEW, and NUMVALID parameters. To use the NUMVIEW and NUMVALID parameters, you must use the WORKREG parameter. You use the NUMVIEW parameter if you need to know the number of pages sharing the view of your central storage address. NUMVALID is the number of pages currently addressable in central storage (accessed). NUMVIEW and NUMVALID could be used to check how effectively you are using IARVSERV shared storage. Pages that are not accessed have not been read or updated by any program.

The following example shows variations in the coding of the *IARR2V* macro.

```
CONTROL  TITLE 'SHARED PAGES'
*****************************************************************
*                                                               *
*         IARR2V IN ASSEMBLER                                   *
*                                                               *
*****************************************************************
         SPACE
CONTROL  AMODE 31                    31-BIT ADDRESSING MODE.
CONTROL  RMODE ANY                   MAY RESIDE ABOVE OR BELOW 16MB.
         SPACE 2
CONTROL  START 0                     DEFINE CSECT NAME AND INITIALIZE
*                                    A LOCATION COUNTER TO ADDRESS 00.
         SPACE
         GBLC  &SYSSPLV              DEFINE THE GLOBAL VARIABLE THAT
*                                    WILL BE SET BY THE SPLEVEL MACRO.
         SPACE
         SPLEVEL TEST                THIS MACRO SETS &SYSSPLV TO:
*                                       1 > ASSEMBLING O/S IS MVS/370.
*                                       2 > ASSEMBLING O/S IS MVS/XA.
*                                       3 > ASSEMBLING O/S IS MVS/ESA.
         TITLE 'STANDARD REGISTER EQUATES'
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         TITLE 'TEST CASE IARR2V IN ASSEMBLER'
SAVEREGS EQU   *
         SPACE
         BAKR  R14,0                 SAVE CALLER'S ARS, GPRS, AND    .
*                                    RETURN ADDRESS ON LINKAGE STACK.
IDENTIFY EQU   *
         SPACE
         USING CONTROL,R15           ESTABLISH TEMPORARY ADDRESABILITY.
         B     SETBEG                BRANCH AROUND IDENTIFIER.
         SPACE
         DC    CL8'CONTROL'          THE NAME OF THIS CSECT.
         DC    CL8'&SYSDATE'         ASSEMBLY DATE.
         DC    CL7'&SYSTIME'         ASSEMBLY TIME.
         DC    CL1'&SYSSPLV'         ASSEMBLING OPERATING SYSTEM.
         SPACE
         DROP  R15
*****************************************************************
*   1.Testcase Mainline Starts Here                             *
*****************************************************************
SETBEG   EQU   *
         LAE   R12,0(R15,0)
         USING CONTROL,R12
*****************************************************************
*                                                               *
*     Run in supervisor state key 0                             *
*                                                               *
*****************************************************************
*****************************************************************
*  2.GETMAIN THE SOURCE  FROM A PAGEABLE SUBPOOL
*****************************************************************
         STORAGE OBTAIN,LENGTH=4096*8,BNDRY=PAGE,SP=0,COND=YES
         ST    1,VSA
         LTR   15,15
         BNZ   NOSTOR
*****************************************************************
         LRA   1,VSA
         LR    5,1
*****************************************************************
*  3.GET A VIRTUAL ADDRESS FROM A REAL ADDRESS
*****************************************************************
INVOKE1  IARR2V RSA=(5),VSA=VSAOUT,LINKAGE=SYSTEM
*
*****************************************************************
*  4.GET THE ADDRESS SPACE
*****************************************************************
INVOKE2  IARR2V RSA=(5),ASID=ASIDO,LINKAGE=SYSTEM
*
*****************************************************************
*  5.GET THE STOKEN FOR THE DATA SPACE
*****************************************************************
INVOKE3  IARR2V RSA=(5),STOKEN=STOKENO,LINKAGE=SYSTEM
*
*****************************************************************
```

```
*  6.GET RETURN CODE AND REASON CODE
*********************************************************************
INVOKE4  IARR2V RSA=(5),RETCODE=RCODE,RSNCODE=RSNC
*
*********************************************************************
*  7.GET THE PAGE SHARING COUNT
*********************************************************************
INVOKE5  IARR2V RSA=(5),WORKREG=(6),NUMVIEW=VIEWS,NUMVALID=VALS
*
*********************************************************************
*  8.GET THE PAGE SHARING COUNT
*********************************************************************
INVOKE6  IARR2V RSA=(5),WORKREG=(6),NUMVIEW=VIEWS,NUMVALID=VALS,       *
               VSA=VSAOUT,ASID=ASIDO,STOKEN=STOKENO,RETCODE=RCODE,     *
               RSNCODE=RSNC
*
*********************************************************************
*  9.GET THE PAGE SHARING COUNT
*********************************************************************
INVOKE7  IARR2V RSA=(5),WORKREG=(6),NUMVIEW=VIEWS,NUMVALID=VALS,       *
               VSA=(9),ASID=(10),STOKEN=STOKENO,RETCODE=RCODE,         *
               RSNCODE=RSNC
*********************************************************************
* The following invocation will get an MNOTE for NUMVIEW and
* NUMVALID since they can no longer be registers but must be
* variables
*********************************************************************
* INVOKE8  IARR2V RSA=(5),WORKREG=(6),NUMVIEW=(7),NUMVALID=(8),
*               VSA=(9),ASID=(10),STOKEN=STOKENO,RETCODE=RCODE,
*               RSNCODE=RSNC
*********************************************************************
NOSTOR   EQU   *
         SAC   0
         SR    R15,R15           set completion code
         PR                      restores registers 2-14, amode,
*                                ascenv, and returns to caller.
*********************************************************************
*  DECLARES
*********************************************************************
VSA      DS    F
VSAOUT   DS    F
ASIDO    DS    F
RCODE    DS    F
RSNC     DS    F
WREG     DS    F
VIEWS    DS    F
VALS     DS    F
STOKENO  DS    2F
*********************************************************************
*                                                                   *
* All generated literals (e.g. =C'YES' or =X'FF' or =F'-1') will be  *
* inserted after the following "LTORG" (LITERAL ORIGIN) statement.   *
*                                                                   *
*********************************************************************
         SPACE
         LTORG
*********************************************************************
*            Include mapping of the VRL                             *
*********************************************************************
CONTROL  CSECT
         SPACE 2
         END   ,                 DENOTES END OF MY PROGRAM.
```

D.3 IPCS RSMDATA Shared Data Report

IPCS has a new subcommand to support shared data RSMDATA SHRDATA. Reports for RSMDATA have been updated where necessary to give shared data information. A sample of the reports follow showing the new or added fields. This report was generated for section D.1, “Sample Program Sharing Data Above and Below 16MB” on page 201.

```
//USERID1I JOB (ACCNT#),
//  CLASS=A,MSGCLASS=X,
//  NOTIFY=USERID1
//IPCS     EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=3M
//SYSPROC DD DSN=SYS1.SBLSCLI0,DISP=SHR
//IPCSDDIR DD   DSN=USERID1.DDIR,DISP=SHR
//DUMP     DD   DSN=USERID1.DUMP.TEST1,DISP=SHR
//IPCSTOC  DD   SYSOUT=*
//IPCSPRNT DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
 IPCS NOPARM
 PROFILE LINESIZE(100) PAGESIZE(80)
 SETDEF DSN('USERID1.DUMP.TEST1') NOCONFIRM PRINT NOTERM
 RSMDATA SHRDATA
/*
```

D.3.1 RSMDATA SHRDATA Report

The *RSMDATA SHRDATA* report is the generated output from IPCS. The report is from a dump generated from the shared data example. See D.1, “Sample Program Sharing Data Above and Below 16MB” on page 201.

```
7.
                          R S M   S H A R E D   D A T A   R E P O R T
SH TOKEN K GP R V P B STAT R LOC LOC       LOC2     PAGE I/O VT O L F D JOBNAME  ASID DSP NAME PAGE     DG       DG
-------- - -- - - - - ---- ----- --------- --------- -------- -- - - - - -------- ---- -------- -------- -------- --------
01C4B480 8 -   E N N N REAL 0E36A -         -         -        SW N N N N USERID1I 001E -        00007000 01C43720 00000000
                                                               SW Y N N N USERID1I 001E -        05702000 01C43220 0E92C000
01C4B840 8 -   E N N N REAL OCC5A -         -         -        SW N N N N USERID1I 001E -        00006000 01C43060 00000000
                                                               SW Y N N N USERID1I 001E -        05701000 01C43560 0E92C000
01C4B900 8 -   E N N N REAL 0F3C0 -         -         -        SW N N N N USERID1I 001E -        00008000 01C437C0 00000000
```

```
                                                                SW Y N N N USERID1I 001E -        05703000 01C431A0 0E92C000
01C488C0 8 -   E N N N REAL 0E3AE -         -         -        SW N N N N USERID1I 001E -        00014000 01C43640 00000000
                                                                SW Y N N N USERID1I 001E -        0570F000 01C432A0 0E92C000
01C48900 8 -   E N N N REAL 0E4B0 -         -         -        SW N N N N USERID1I 001E -        00013000 01C43120 00000000
                                                                SW Y N N N USERID1I 001E -        0570E000 01C43C00 0E92C000
01C48A20 8 -   E N N N REAL 0E9BB -         -         -        SW N N N N USERID1I 001E -        00010000 01C43920 00000000
                                                                SW Y N N N USERID1I 001E -        0570B000 01C43CA0 0E92C000
01C48A60 8 -   E N N N REAL 0E2AD -         -         -        SW N N N N USERID1I 001E -        00015000 01C43E40 00000000
                                                                SW Y N N N USERID1I 001E -        05710000 01C43460 0E92C000
01C48BC0 8 -   E N N N REAL 0E46D -         -         -        SW N N N N USERID1I 001E -        00009000 01C43660 00000000
                                                                SW Y N N N USERID1I 001E -        05704000 01C43CE0 0E92C000
01C48C00 8 -   E N N N REAL 0481E -         -         -        SW N N N N USERID1I 001E -        0000C000 01C43300 00000000
                                                                SW Y N N N USERID1I 001E -        05707000 01C43820 0E92C000
01C48CE0 8 -   E N N N REAL 0E21F -         -         -        SW N N N N USERID1I 001E -        0000D000 01C43DC0 00000000
                                                                SW Y N N N USERID1I 001E -        05708000 01C43C20 0E92C000
01C48D00 8 -   E N N N REAL 0E9BA -         -         -        SW N N N N USERID1I 001E -        0000F000 01C43C60 00000000
                                                                SW Y N N N USERID1I 001E -        0570A000 01C436E0 0E92C000
01C48D80 8 -   E N N N REAL 0EA20 -         -         -        SW N N N N USERID1I 001E -        0000B000 01C435A0 00000000
                                                                SW Y N N N USERID1I 001E -        05706000 01C43860 0E92C000
01C48DE0 8 -   E N N N REAL 0E21E -         -         -        SW N N N N USERID1I 001E -        0000E000 01C43700 00000000
                                                                SW Y N N N USERID1I 001E -        05709000 01C43140 0E92C000
01C48E20 8 -   E N N N REAL 0E269 -         -         -        SW N N N N USERID1I 001E -        0000A000 01C43980 00000000
                                                                SW Y N N N USERID1I 001E -        05705000 01C43F60 0E92C000
01C48EA0 8 -   E N N N REAL 0E3AF -         -         -        SW N N N N USERID1I 001E -        00011000 01C43320 00000000
                                                                SW Y N N N USERID1I 001E -        0570C000 01C43FC0 0E92C000
01C48F40 8 -   E N N N REAL 0E4B1 -         -         -        SW N N N N USERID1I 001E -        00012000 01C430C0 00000000
                                                                SW Y N N N USERID1I 001E -        0570D000 01C43540 0E92C000

Totals (in decimal):
           REAL            EXP            AUX            DSN
             16              0              0              0
           FREF           DREF            DIV          TOTAL
              0              0              0             16
```

Appendix E. Code Examples for APPC/MVS Enhancements

The following examples show the code needed to link to an error analysis routine, and how to use the error extract services.

E.1 Example Used to Call for CPI Communications

```
/*********************************************************************/
/*                      CPICOMM routine                              */
/* This subroutine is used to issue all CPIC requests. We use it     */
/* to test the REXX return code from the address cpicomm statement   */
/* here.  All cpicomm returned indicators are tested in the caller's */
/* routine.                                                          */
/* Input:                                                            */
/* -string to be executed as a CPIC command is passed as an argument */
/* Output:                                                           */
/* -none                                                             */
/*                                                                   */
/*********************************************************************/
cpicomm:

address CPICOMM arg(1)
trace ?r                                                  /*HDM*/
invalid_rc = RC
function = word(arg(1),1)
j = words(arg(1))                 /* get number of parm's */  /*HDM*/
cpic_return_code = value(word(arg(1),j)) /*..... */          /*HDM*/
                                /* the last is the CPIC rtnc*/ /*HDM*/
if cpic_return_code <> 0 then do                          /*HDM*/
  say 'Address CPICOMM 'function' returned bad return code',     /*HDM*/
         cpic_return_code                                       /*HDM*/
       say 'If you want to know details answer Yes'             /*HDM*/
       Pull answer                                              /*HDM*/
       if answer = 'Y' then do /* go and ask ATBEES3 for details  HDM*/
         /* word(2) contains the conv_id, mind the VALUE          HDM*/
         help = value(word(arg(1),2))                           /*HDM*/
         Call EES3(help)       /* Call the ATBEES3 i/f rtn */   /*HDM*/
       end /* detail analysis */                                /*HDM*/
  end                                                           /*HDM*/

if invalid_rc = 0 then return
/* in TSO RC is set to the contents of return_code  */
/* therefore we only say RC if it is outside the    */
/* range of return_codes returned by CPI-CI         */
if invalid_rc < 0 | invalid_rc > 34 then do
   say '>> A return code = ' invalid_rc ' occured when trying to'
   say '>> execute the ' function ' CPIC function.'
end
return
```

E.2 Sample Code to Call ATBEES3

```
/* Rexx , why do we need it to tell. CLISTs should be marked*/
EES3:
Arg conv_id .
Say 'Extracting Error information for Conversion ID' , c2x(conv_id)

Service_name = '        '
Service_reason_code = '00000000'x
Message_text_length = '00000000'x
Message_text = Copies(' ',512)
Error_log_product_set_ID_length = '00000000'x
Error_log_product_set_ID_= Copies(' ',256)
Error_log_information_length = '00000000'x
Error_log_information = Copies(' ',512)

Reason_code  = '00000004'x
Return_code  = '00000008'x
Address LINKPGM 'ATBEES3 conv_id ',
        'Service_name Service_reason_code ',
        'Message_text_length Message_text ',
        'Error_log_product_set_ID_length ' ,
        'Error_log_product_set_ID ',
        'Error_log_information_length ',
        'Error_log_information ',
        'Reason_code Return_code'
If return_code <> '00000000'x then do
  Say 'Cannot extract error information'
  Say 'ATBEES3 returns ' c2x(return_code) 'Reason code ',
       c2x(reason_code)
  end
else do
say 'Service which caused caused error ' Service_name ,
    'Reason Code ' c2x(Service_reason_code)
  if Message_text_length > '00000000'x then do
    message_text = strip(message_text)
    i = words(message_text)
    m = 1
    do while j < i
    line = ''
    do j = m to i while length(line) < 78
      line =line' 'word(message_text,j)
      end
    say line
    m = j
    end /* while */
    say ' '
  end /* Message available */
end /* ATBEES3 returned without an error */

return
```

Appendix F. OpenEdition MVS

This appendix describes many of the installation requirements for installing OpenEdition MVS.

F.1 OpenEdition MVS DASD Installation Requirements

Table 8. DASD Storage Requirements for OpenEdition MVS

DDDEF	BLK SIZE	No of BLKS	No of 3380 TRKS	No of 3390 TRKS	No of DIR BLKS
CMDLIB	6144	127	19	17	5
CSSLIB	6144	129	19	17	33
HELP	6160	74	11	10	6
LINKLIB	6144	1854	266	233	48
LPALIB	6144	125	18	18	5
MACLIB	6160	225	33	29	5
MIGLIB	6144	142	21	18	12
MSGENU	6160	18	3	3	2
NUCLEUS	6144	17	3	3	2
PARMLIB	6160	17	3	3	2
PROCLIB	6160	15	3	3	2
SAMPLIB	6160	219	33	31	11
SBLSPNL0	6160	99	15	13	7
SBPXEXEC	6160	77	11	10	3
SBPXMENU	6160	61	9	8	7
SBPXPENU	6160	175	25	22	19
SBPXTENU	6160	16	3	3	2
SCEELKED	6144	125	18	16	22
SCEERUN	6144	111	16	14	3
SFOMHDRS	6160	69	12	9	6

Note

It is advisable to allow some additional space for future maintenance and updates.

F.2 OpenEdition MVS Sample BPXPRMxx Parmlib Member

```
MAXPROCSYS(300)                 /* System will allow at most 200
                                   processes to be active
                                   concurrently                 */

MAXPROCUSER(25)                 /* Allow each user (same UID) to
                                   have at most 25 concurrent
                                   processes active             */

MAXUIDS(200)                    /* Allow at most 200 OpenMVS users
                                   to be active concurrently    */

MAXFILEPROC(64)                 /* Allow at most 64 open files
                                   per user                     */

MAXPTYS(256)                    /* Allow up to 256 pseudo-terminal
                                   sessions                     */

CTRACE(CTIBPX00)                /* Parmlib member 'CTIBPX00' will
                                   contain the initial tracing
                                   options to be used           */

STEPLIBLIST('/system/steplib')  /* HFS file /system/steplib will
                                   contain the list of sanctioned
                                   step libraries for set-user-ID
                                   and set-group-ID executables. */

FILESYSTYPE TYPE(HFS)
            ENTRYPOINT(GFUAINIT)
            PARM('')

ROOT     FILESYSTEM('OMVS.ROOT.HFS')
         TYPE(HFS)
         MODE(RDWR)

MOUNT    FILESYSTEM('MARTIN.HFS')
         TYPE(HFS)
         MOUNTPOINT('/u/martin')
         MODE(RDWR)

FILESYSTYPE TYPE(INET) ENTRYPOINT(BPXTIINT)
NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(64)
        TYPE(INET)

FILESYSTYPE TYPE(IBMUDS) ENTRYPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
        DOMAINNUMBER(1)
        MAXSOCKETS(64)
        TYPE(IBMUDS)

MAXTHREADTASKS(50)              /* System will allow 50 threads
                                   to be active concurrently    */

MAXTHREADS(200)                 /* System will allow at most 200
                                   threads to be active         */
```

F.3 OpenEdition MVS Sample ISPF Selection Panel

```
%----------------------  ISPF/PDF PRIMARY OPTION MENU ---------
%OPTION  ===>_ZCMD
%
%   0 +ISPF PARMS  - Specify terminal and user parameters
%   1 +BROWSE      - Display source data or output listings
%   2 +EDIT        - Create or change source data
%   3 +UTILITIES   - Perform utility functions
%   4 +FOREGROUND  - Invoke language processors in foreground
%   5 +BATCH       - Submit job for language processing
%   6 +COMMAND     - Enter TSO Command, CLIST, or REXX exec
%   7 +DIALOG TEST - Perform dialog testing
%   8 +LM UTILITIES- Perform library administrator utility functions
%   9 +IBM PRODUCTS- Additional IBM program development products
%   X +EXIT        - Terminate ISPF using log and list defaults


%  1F - Browse files
%  2F - Edit files
% ISH - OpenMVS ISPF Shell


)INIT
  HELP = ISR00003
)PROC
  IF (&ZCMD ¬= ' ')
    &ZQ = TRUNC(&ZCMD,'.')
    IF (&ZQ = ' ')
      .MSG = ISRU000
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                0, 'PANEL(ISPOPTA)'
               1F,'CMD(OBROWSE)'
               2F,'CMD(OEDIT)'
              ISH,'CMD(ISHELL)'
              ' ',' '
                *,'?' )
)END
```

Figure 83. Sample OpenEdition MVS ISPF/PDF Selection Menu

You must make the following changes to an ISPF/PDF selection panel:

1. Add a statement to the list of options for Browse files. Be sure to include a selection number with the statement. In Figure 83 this is:

   ```
   %  1F - Browse files
   ```

2. Add a statement to the)PROC section of the panel to invoke OBROWSE. In Figure 83, this is:

   ```
   1F,'CMD(OBROWSE)'
   ```

   Be sure that the symbol at the start of this statement (1F in Figure 83) matches the number specified in the list of options.

3. Add a statement to the list of options for edit files. Be sure to include a selection number with the statement. In Figure 83, this is:

```
% 2F - Edit files
```

4. Add a statement to the)PROC section of the panel to invoke OEDIT. In Figure 83 on page 213, this is:

   ```
   2F,'CMD(OEDIT)'
   ```

   Be sure that the symbol at the start of this statement (2F in Figure 83 on page 213) matches the number specified in the list of options.

5. Add a statement to the list of options for OpenMVS ISPF Shell. Be sure to include a selection number with the statement. In Figure 83 on page 213, this is:

   ```
   % ISH - OpenMVS ISPF Shell
   ```

6. Add a statement to the)PROC section of the panel to invoke the OpenMVS ISPF Shell environment. In Figure 83 on page 213, this is:

   ```
   ISH,'CMD(ISHELL)'
   ```

   Be sure that the symbol at the start of this statement (ISH in Figure 83 on page 213) matches the number specified in the list of options.

F.4 OpenEdition MVS Managing Multiple File Systems

This figure shows the recommended approach to managing multiple HFSs in an OpenEdition MVS environment.

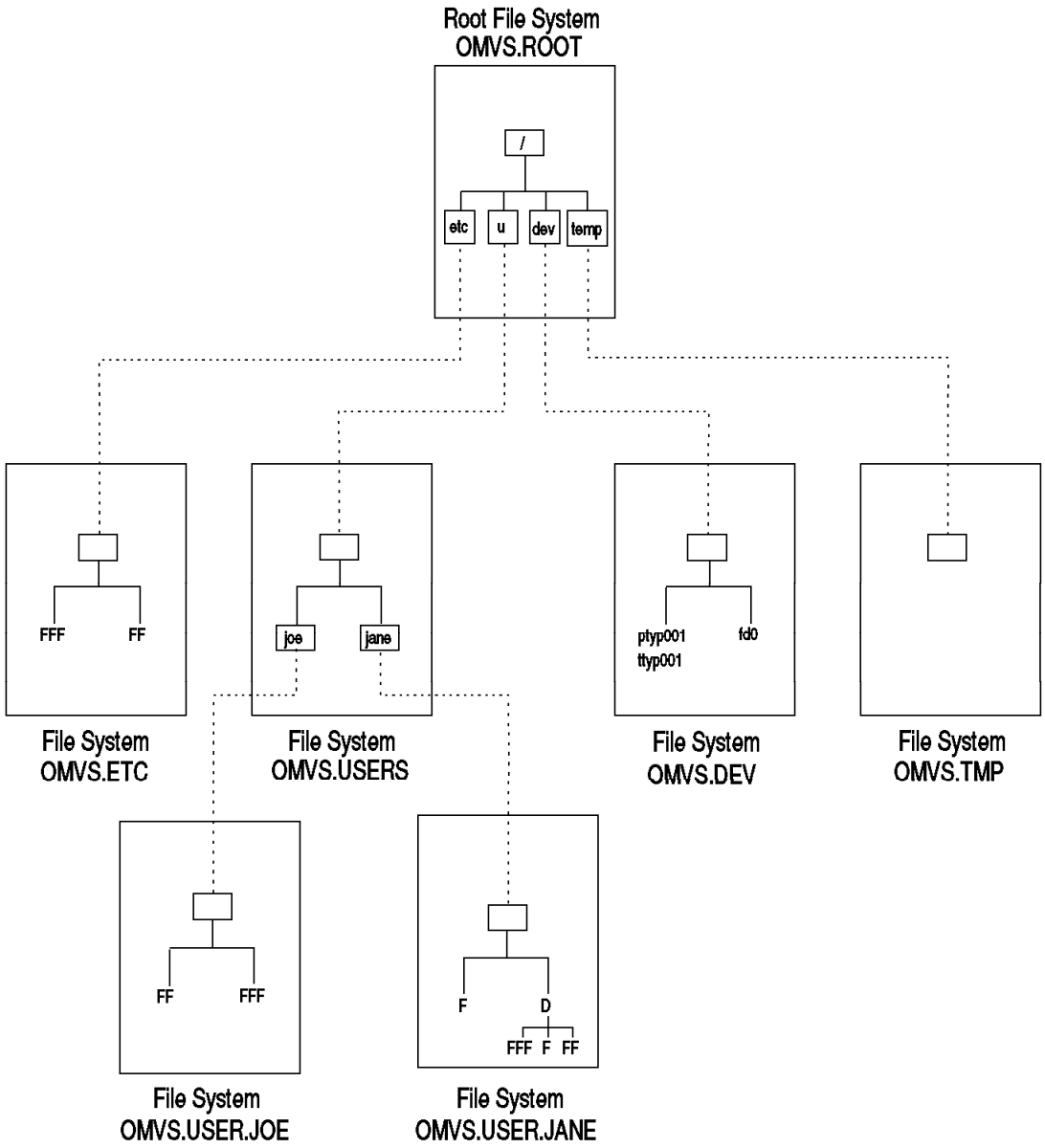


Figure 84. Recommended OpenEdition MVS File System Setup

F.5 OpenEdition MVS Sample Socket Application

F.5.1.1 Server Part

```
/* This is part 1 of a simple socket application.  This
 * program, the server, opens a socket and listens for an
 * incoming connection.
 *
 * Once a connection is made to the client program, data
 * is received and transmitted back to the client.
 * If all is successful, the socket will be closed and the
 * application terminates.
 *
 * The program will print descriptive information along with
 * any error messages.
 *
 * Once the server.c code has been compiled and linked, the
 * application can be started by entering:
 *
 * server <socket number> <# of bytes to receive and transmit>
 * eg: server 2001 16
 *
 * The code was kindly supplied by:
 *
 * Open Software Associates Ltd.
 * P.O.Box 401
 * Ringwood
 * VICTORIA
 * AUSTRALIA 3134
 * tel: +61-3-871-1666 fax: +61-3-879-4696
 *
 *                           DISCLAIMER
 *
 * This source code has not been submitted to a formal test by IBM or
 * Open Software Associates Ltd. 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 and Open Software Associates Ltd for accuracy, 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.
 *
*/
#define _INCLUDE_HPUX_SOURCE 1
#include <time.h>
#if defined(__unix)
#endif
#pragma map(close_socket,"@@CLOSE")
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#if !defined(MSW)
#define far
#define ERRNODEFN extern
#endif
#if defined(MSW)
#define ERRNODEFN
#endif
#if defined(__unix)
#define close_socket(a) close(a)
#endif
#define BUFFER_SIZE 1000
char sendbuf[BUFFER_SIZE];
char recvbuf[BUFFER_SIZE];
#if 0
#define INADDR_ANY              (u_long)0x00000000
struct in_addr {
        u_long s_addr;
};
struct sockaddr_in {
        u_char  sin_len;
        u_char  sin_family;
        u_short sin_port;
        struct  in_addr sin_addr;
char sin_zero[8];
};
#endif
struct sockaddr_in server_addr, client_addr;
ERRNODEFN int errno;
int debug = 1;
main(int argc, char *argv[])
 int ret,
  i,
  len,
  total,
  listen_sd,
  acc_sd;
unsigned int size;
if (argc < 3)
{
 fprintf(stderr, "Usage: %s <local_port> <xfer_size>\n",
  argv[0]);
 exit(1);
}
size = atoi(argv[2]);
/*
 *     Create a socket locally to serve as a communication
 *     endpoint.
 */
listen_sd = socket(AF_INET, SOCK_STREAM, 0);
if (listen_sd < 0)
{
 fprintf(stderr, "Error: socket call failed - errno=%d\n",
  errno);
 fprintf(stderr, "\n%s", strerror(errno));
 exit(1);
}
if (debug)
 fprintf(stderr, "socket() returned %d\n", listen_sd);
/*
 * 'Bind' the newly created socket to a wildcard local IP
 * address and the local TCP port specified on the command line.
 */
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(atoi(argv[1]));
server_addr.sin_addr.s_addr = INADDR_ANY;
ret = bind(listen_sd, (struct sockaddr far *)&server_addr,
 sizeof(server_addr));
if (ret < 0)
{
 fprintf(stderr, "Error: bind call failed errno = %d\n",
  errno);
 close_socket(listen_sd);
 exit(1);
}
if (debug)
 fprintf(stderr, "bind() returned %d\n", ret);
/*
 *     Create a queue of length one to hold incoming connection
 *     requests.
 */
if (debug)
 fprintf(stderr, "Server will listen on port %d\n",
  ntohs(server_addr.sin_port));
ret = listen(listen_sd, 1);
if (ret < 0)
{
 fprintf(stderr, "Error: listen call failed - errno = %d\n",
  errno);
 close_socket(listen_sd);
 exit(1);
}
if (debug)
 fprintf(stderr, "listen returned %d\n", ret);
/*
 * Call accept which will block until an incoming connection
 * is established on the listening socket.  Accept() then returns
 * the socket descriptor of the new socket and will fill in the
 * 'client_addr' socket address structure with the remote IP
 * address and port.
 */
len = sizeof(struct sockaddr_in);
if (debug)
 fprintf(stderr,
  "blocking on accept() for an incoming connection ..\n");
acc_sd = accept(listen_sd, (struct sockaddr far *)&client_addr, &len);
if (acc_sd < 0)
{
 fprintf(stderr, "Error: accept call failed - errno = %d\n",
  errno);
 close_socket(listen_sd);
 exit(1);
}
if (debug)
 fprintf(stderr, "Established connection with system 0x%lx\n",
  client_addr.sin_addr);
/*
 * The connection is now Established.
 *
 * We will receive all of the data sent by the remote node
```

```
 * before sending data out.  There is one caveat, however.
 * The recv() call is defined such that it returns as
 * soon as ANY data is available rather than waiting until
 * all 'size' bytes of data have been read.
 * Therefore, we must loop on the recv() call until all
 * the data has been received.
 */
total = 0;
if (debug)
 fprintf(stderr, "Now attempting to receive data ...\n");
do
{
 len = recv(acc_sd, (char far *)&recvbuf[total], size - total,
  0);
 if (len <= 0)
 {
  fprintf(stderr,
   "Error: recv() call failed - errno %d\n",
   errno);
  close_socket(listen_sd);
  exit(1);
 }
 if (debug)
  fprintf(stderr, "recv() returned %d bytes\n", len);
 total += len;
} while (total < size);
/*
 * Echo the received data back to the remote node.
 */
len = send(acc_sd, (char far *)recvbuf, size, 0);
if (len < 0)
{
 fprintf(stderr, "Error: send() call failed - errno %d\n",
  errno);
 close_socket(listen_sd);
 close_socket(acc_sd);
 exit(1);
}
/*
 * All data has been received and transmitted, so close the
 * sockets and exit.
 */
fprintf(stderr, "All data transferred closing socket...\n");
close_socket(acc_sd);
close_socket(listen_sd);
exit(0);
}
```

F.5.1.2 Client Part

```
/* This is part 2 of a simple socket application.
 * This program, the client, connects to a remote server,
 * sends, and then receives data back from the server.
 *
 * The program will print descriptive information along with
 * any error messages.
 *
 * Once the client.c code has been compiled and linked, the
 * application can be started by entering:
 *
 * client <addr server> <socket#> <# of bytes> <delay>
 * eg: client 9.12.13.69 2001 16 2
 *
 * The code was kindly supplied by:
 *
 * Open Software Associates Ltd.
 * P.O.Box 401
 * Ringwood
 * VICTORIA
 * AUSTRALIA 3134
 * tel: +61-3-871-1666 fax: +61-3-879-4696
 */
#define _INCLUDE_HPUX_SOURCE 1
#include <time.h>
#if defined(__unix)
#endif
#pragma map(close_socket,"@@CLOSE")
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#if !defined(MSW)
#define far
#define ERRNODEFN extern
#endif
#if defined(MSW)
#define ERRNODEFN
#endif
#if defined(__unix)
#define close_socket(a) close(a)
#endif
#define BUFFER_SIZE 1000
#define LINE_LEN 70
char sendbuf[BUFFER_SIZE];
char recvbuf[BUFFER_SIZE];
struct sockaddr_in server_addr, client_addr;
ERRNODEFN int errno;
int debug = 1;
#if defined(MSW)
int
sleep(int seconds)
{
 long start_seconds,
  current_seconds,
  end_seconds;
 time(&start_seconds);
 end_seconds = start_seconds + (long)seconds;
 do
 {
  time(&current_seconds);
 } while (current_seconds < end_seconds);
 return 0;
}
#endif /* MSW */

main(int argc, char *argv[])
{
 int ret,
  i,
  len,
  total,
  size,
  sd,
  errno,
  seconds;
 if (argc < 5)
 {
  fprintf(stderr,
   "Usage: %s <faddr> <fport> <xfer_size> <delay>\n",
   argv[0]);
  exit(1);
 }
 size = atoi(argv[3]);
 /*
  * Create a local endpoint for communication.
  */
 sd = socket(AF_INET, SOCK_STREAM, 0);
 if (sd < 0)
 {
  fprintf(stderr, "Error: socket call failed - errno=%d\n",
   errno);
  fprintf(stderr, "\n%s", strerror(errno));
  exit(1);
 }
 if (debug)
  fprintf(stderr, "socket() returned %d\n", sd);
 /*
  * Fill in a socket address structure with the necessary
  * information about the remote server node (remote node IP
  * address and port for incoming connections) and attempt to
  * connect to the server.  This connect will block until the
  * remote server has accepted the connection or the
  * connection request times out.
  */
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(atoi(argv[2]));
 server_addr.sin_addr.s_addr = inet_addr(argv[1]);
 ret = connect(sd, (struct sockaddr far *)&server_addr,
  sizeof(server_addr));
 if (ret < 0)
 {
  fprintf(stderr, "Error: connect() call failed errno = %d\n",
   errno);
  fprintf(stderr, "\n%s", strerror(errno));
  close_socket(sd);
  exit(1);
 }
 if (debug)
  fprintf(stderr, "Established connection with system 0x%lx\n",
   server_addr.sin_addr);
 seconds = atoi(argv[4]);
 if (debug)
  fprintf(stderr, "Waiting %d seconds before sending data ...\n",
   seconds);
 sleep(seconds);
 /*
  * Send the number of bytes specified on the command line to
  * the server.
  */
 for (i = 0; i < size; i++)
```

```
 sendbuf[i] = (i % 26) + 'a';
len = send(sd, (char far *)sendbuf, size, 0);
/*
 * After the server has received all of the data, it will
 * send it back to us.  As described in the server code
 * comments, we must loop on rev() calls until all the data
 * has been received.
 */
total = 0;
if (debug)
 fprintf(stderr, "Now attempting to receive data ...\n");
do
{
 len = recv(sd, (char far *)&recvbuf[total], size - total, 0);
 if (len <= 0)
 {
  fprintf(stderr,
   "Error: recv() call failed - errno %d\n",
   errno);
 fprintf(stderr, "\n%s", strerror(errno));
  close_socket(sd);
  exit(1);
 }
 if (debug)
  fprintf(stderr, "recv() returned %d bytes\n", len);
 total += len;
} while (total < size);
if (debug)
{
 fprintf(stderr, "recvbuf contains:\n");
 for (i = 0; i < len; i++)
 {
  fprintf(stderr, "%c", recvbuf[i]);
  if (i % LINE_LEN == LINE_LEN - 1)
   fprintf(stderr, "\n");
 }
 fprintf(stderr, "\n");
}
/*
 * All the data has been transferred.  Close the socket and exit.
 */
fprintf(stderr, "All data transferred closing socket...\n");
close_socket(sd);
exit(0);
}
```

F.6 Internationalization

```
/* Example how to handle locales */
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <env.h>
#include <locale.h>
#include <time.h>

void Upper   ( char * string  ) ;
void Show_LC ( void ) ;
char *countries[] = { "USA",
                      "UK" ,
                      "GERM",
                      NULL } ;  /* terminate list with null ptr */
/* Effectively it is NOT necessary to have different formats for
   strftime to display time according to country specifics.  This
   is just in case one wants add nice text like
   .... Guten Tag ....  This problem is not covered by
   Internationalization.
*/
char *time_format[] = { "%x %X",
                        "%c" ,
                        "%x %X",
                      NULL } ;  /* terminate list with null ptr */

char *time_zone[] = { "EDT5EST",
                      "GMTOGMT" ,
                      "MEZ-1MESZ-2",
                      NULL } ;  /* terminate list with null ptr */

int main(int argc, char *argv[], char *env[])
{ int i,j,k,lm,n;
  char *string;
  char parm[16];
  char dest[96];
  time_t temp ;
  struct tm *timeptr;

  printf("\ntzname[0]=%s, tzname[1]=%s \n", tzname[0], tzname[1]);
  printf("\nThe original values were...\n");
  printf("\nCurrent time zone setting : %s",  getenv("TZ"));
  if ( (string = getenv("LC_ALL")) != 0) {
    printf("\nLC_ALL from Environment : %s", string);
    }
  Show_LC();

  if (argc > 1) {
    printf("\nParameter received %s",    argv[1] );
    strncpy(parm,argv[1],4); Upper(parm);
    if ((i=locate_country(parm)) >=0) {
      time(&temp) ;
      timeptr = localtime(&temp);
      j = strftime(dest,sizeof(dest)-1, time_format[i],timeptr);
      printf("\n%s", dest);
      printf("\nGoing to change locales according to %s", parm);
      setenv("TZ", time_zone[i],1);
      tzset();
      printf("\ntzname[0]=%s, tzname[1]=%s \n", tzname[0], tzname[1]);
      setenv("LC_ALL", parm, 1);
      setlocale(LC_ALL,"");
      printf("\nCurrent time zone setting : %s",  getenv("TZ"));
      Show_LC();
      time(&temp) ;
      timeptr = localtime(&temp);
      j = strftime(dest,sizeof(dest)-1, time_format[i],timeptr);
      printf("\n%s", dest);
    }
    else {
      printf("\nCountry %s not in list\n", parm);
      exit(-1) ;
    }
  } else printf("\nNo parameter given. Nothing changed.\n");
printf("\n");
exit(0);
}

void Upper  (char *string)   /* strupr is not POSIX */
{ char *z;
  z = string ;
  while (*z) {
    *z = toupper((int)*z)  ;
    z++;
  }
}

int  locate_country (char *string)
{ int i=0 ;

  while(countries[i]) {
  if (!(strcmp(countries[i],string))) return(i);
  i++;
  } /* end while */
  return(-1);
}

void Show_LC (void)
{ char *string;
  string = setlocale(LC_COLLATE , NULL); /* Tell me... */
  printf("\nLC_COLLATE:  %s", string);
  string = setlocale(LC_CTYPE, NULL); /* Tell me... */
  printf("\nLC_CTYPE:    %s", string);
  string = setlocale(LC_MONETARY, NULL); /* Tell me... */
  printf("\nLC_MONETARY: %s", string);
  string = setlocale(LC_NUMERIC, NULL); /* Tell me... */
  printf("\nLC_NUMERIC:  %s", string);
  string = setlocale(LC_TIME, NULL); /* Tell me... */
  printf("\nLC_TIME:     %s", string);
  string = setlocale(LC_TOD, NULL); /* Tell me... */
  printf("\nLC_TOD:      %s", string);
  printf("\n");
}
```

Index

Special Characters

Numerics

A

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
MVS/ESA SP Version 5
Implementation Guide
Version 5.1.0
Version 5.2.0
November 1995

Publication No. SG24-4584-00

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?

_____

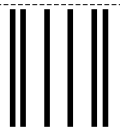
Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____ _____
Name Address

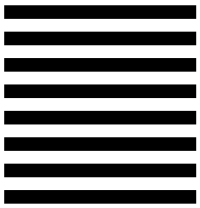
_____ _____
Company or Organization

_____ _____
Phone No.

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

Printed in U.S.A.