GC28-1143-6
File No. S370-34

Program Product

MVS/Extended Architecture Conversion Notebook Volume 1

MVS/SP Version 2 Release 2 and Related Products

JES3 Version 2	**5665-291**
JES2 Version 2	**5740-XC6**

IBM

Seventh Edition (June, 1987)

This is a major revision of GC28-1143-5. See the Summary of Amendments following the Contents for a summary of the changes made to this manual.

This edition applies to the following program products until otherwise indicated in new editions or technical newsletters:

Assembler H Version 2 (5668-962)
MVS/XA Data Facility Product (MVS/XA DFP) Version 2 Release 3 (5665-XA2)
MVS/System Product - JES2 (MVS/SP) Version 2 Release 2 (5740-XC6)
MVS/System Product - JES3 (MVS/SP) Version 2 Release 2 (5665-291)

The previous edition may now be ordered using the temporary number, GT00-2111. It still applies to:

Assembler H Version 2 (5668-962)
MVS/XA Data Facility Product (MVS/XA DFP) Version 2 Releases 1,2 (5665-XA2)
MVS/System Product - JES2 (MVS/SP) Version 2 Release 1 (5740-XC6)
MVS/System Product - JES3 (MVS/SP) Version 2 Release 1 (5665-291)

Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System Bibliography*, GC20-0001, for the editions and technical newsletters that are applicable and current.

References in this publication to IBM products 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 in this publication is not intended to state or imply that only IBM's product may be used. Any functionally equivalent product may be used instead.

Publications are not stocked at the address given below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your country.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications Development, Department D58, Building 921, PO Box 390, Poughkeepsie, New York 12602. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

ABOUT THIS BOOK

This book, the *MVS/XA Conversion Notebook, Volume 1* contains very little new information. Rather, it focuses on the history of those changes to MVS/XA that occurred with each level of MVS/SP Version 2 Release 1 that are still relevant for users of MVS/SP Version 2 Release 2.

Installations currently running the last level of Release 1, MVS/SP Version 2 Release 1.7, only need to read the passages marked by change bars in the left margin. Installations currently running an early level of MVS or MVS/XA and planning to convert to MVS/SP Version 2 Release 2 must read the passages that are new to them.

The companion to this book, the *MVS/XA Conversion Notebook, Volume 2* (GC28-1411), describes the updates and enhancements that are new with Release 2. All installations planning to convert to MVS/SP Version 2 Release 2, regardless of which release they are currently running, must read Volume 2.

Even though this book contains little new technical information, it differs from previous editions of the *MVS/XA Conversion Notebook*. Information that no longer applies because of Release 2 updates and enhancements has been deleted. For example, because Releases 2 replaces the IOGEN process with a new procedure for defining I/O devices, the history of changes to IOGEN has been removed from this book. On the other hand, the information about extended storage, included as of Release 2.1.3, remains in the book because it still applies. This information is for installations that are converting from a pre-Release 2.1.3 level of MVS who have not yet learned about extended storage.

Installations that are not converting to a level of MVS/SP Version 2 Release 2, but to an earlier level of MVS/XA, need a previous edition of the *MVS/XA Conversion Notebook*.

Use the following chart to determine which book you need.

Level of the target Base Control Program	**Appropriate Conversion Publications**
MVS/SP Version 2 Release 1.3	*MVS/XA Conversion Notebook* **GC28-1143-3 (orderable as GT00-1785)**
MVS/SP Version 2 Release 1.7	*MVS/XA Conversion Notebook* **GC28-1143-5 (orderable as GT00-2111)**
MVS/SP Version 2 Release 2.0	*MVS/XA Conversion Notebook, Volume 1* **GC28-1143-6 and** *MVS/XA Conversion Notebook, Volume 2* **GC28-1411**
MVS/SP Version 2 Release 2.1	*MVS/XA Conversion Notebook, Volume 1* **GC28-1143-6 and** *MVS/XA Conversion Notebook, Volume 2* **GC28-1411**

Preface

This book is intended for experienced system programmers responsible for converting to any level of MVS/SP Version 2 Release 2 from:

- OS/VS2 MVS Release 3.8 system with at least one of the following installed:
  - For JES2 users, MVS/SP Version 1 Release 3.0 or a later release
  - For JES3 users, MVS/SP Version 1 Release 3.1 or a later release

  Although it is possible to convert other releases of MVS/370 to MVS/XA, this book does not describe how to do so.

- Earlier releases of MVS/XA that include:
  - MVS/SP Version 2 Releases 1.0 through 1.7 and
  - MVS/XA Data Facility Product (MVS/XA DFP) Version 1 or 2

If you are converting from MVS/370, all of this book applies to you. If you are converting from an early release of MVS/XA, read the portions that are new to you. If you are converting from MVS/SP Version 2 Release 1.7, read the Summary of Amendments and the passages marked by a vertical line in the left margin. Such lines indicate areas that have been updated since the last edition of the book.

Regardless of the release from which you are converting, you also need to read the *MVS/XA Conversion Notebook, Volume 2*. This second volume describes the enhancements and updates included in MVS/SP Version 2 Release 2, your target system.

Readers are expected to have an in-depth knowledge of MVS/370, the configuration and procedures of the current installation, and the configuration of the target installation. You also need to be familiar with the MVS/XA overview information in the Licensed Programming Announcement letters and General Information Manuals for MVS/SP Version 2 and MVS/XA DFP Version 2. Reading the *MVS/XA Overview* is also helpful.

Required Program Products

This book contains conversion information related to the program products required for MVS/XA:

- MVS/System Product - JES2 (MVS/SP) Version 2 (5740-XC6)
- MVS/System Product - JES3 (MVS/SP) Version 2 (5665-291)
- MVS/XA Data Facility Product (MVS/XA DFP) Version 2 (5665-XA2)
- Assembler H Version 2 (5668-962)

Naming Conventions

Frequently, the book refers to different levels of MVS/SP Version 2 Release 1 and MVS/XA DFP by the MVS/SP version/release/level number only. Figure 1-1 on page vii explains these references.

Reference	Meaning
Release 2.1.0	MVS/SP Version 2 Release 1.0 and MVS/XA DFP Version 1 Release 1.0
Release 2.1.1	MVS/SP Version 2 Release 1.1 and MVS/XA DFP Version 1 Release 1.1
Release 2.1.2	MVS/SP Version 2 Release 1.2 and MVS/XA DFP Version 1 Release 1.2 or MVS/XA DFP Version 2 Release 1.0 or MVS/XA DFP Version 2 Release 2.0
Release 2.1.3	MVS/SP Version 2 Release 1.3 and MVS/XA DFP Version 1 Release 1.2 or MVS/XA DFP Version 2 Release 1.0 or MVS/XA DFP Version 2 Release 2.0
Release 2.1.3VFE	MVS/SP Version 2 Release 1.3VFE and MVS/XA DFP Version 1 Release 1.2 or MVS/XA DFP Version 2 Release 1.0 or MVS/XA DFP Version 2 Release 2.0
Release 2.1.3AE	MVS/SP Version 2 Release 1.3AE and MVS/XA DFP Version 2 Release 1.0 or MVS/XA DFP Version 2 Release 2.0
Release 2.1.5	MVS/SP Version 2 Release 1.5 and MVS/XA DFP Version 1 Release 1.2 or MVS/XA DFP Version 2 Release 1.0 or MVS/XA DFP Version 2 Release 2.0
Release 2.1.7	MVS/SP Version 2 Release 1.7 and MVS/XA DFP Version 1 Release 1.2 or MVS/XA DFP Version 2 Release 1.0 or MVS/XA DFP Version 2 Release 2.0

Figure 1-1. References to MVS/SP Version 2 and MVS/XA DFP Version 1 and 2 Releases

Additionally, this book uses the term MVS/XA interchangeably with MVS/SP Version 2. It uses the term MVS/370 to refer to OS/VS2 MVS Release 3.8 or any level of MVS/SP Version 1. In discussions that pertain to all levels of a release the book uses an x in place of a level number. Thus, Release 2.1.x refers to any level of MVS/SP Version 2 Release 1; for example, Release 2.1.7 or Release 2.1.3. Release 2.2.x refers to any level of MVS/SP Version 2 Release 2 such as Release 2.2.0.

How this Book is Organized

The information in this book is organized as follows:

"Chapter 1: Introduction" summarizes the work required to convert to MVS/XA and lists conversion tasks you can do on your MVS/370 system before receiving MVS/XA.

"Chapter 2: Installation and Initialization" includes information related to installing and initializing an MVS/XA system and generating the stand-alone dump program.

"Chapter 3: Programming Considerations" describes changes that might affect user-written code, including changes to assembler language instructions and macros. It also describes new function available to programmers.

"Chapter 4: Operating Considerations" describes new and changed commands and operational procedures.

"Chapter 5: System Modifications" describes new and changed user exits and ways of tailoring the system.

"Chapter 6: Problem Determination" describes new and changed ways of tailoring and suppressing dumps, new and changed dump formats, trace facilities, and debugging considerations.

"Chapter 7: Accounting" describes changes that might affect your accounting procedures.

"Chapter 8: Measurement and Tuning" describes changes related to performance.

"Chapter 9: Coexistence Considerations" contains considerations for running both MVS/370 and MVS/XA in a single installation.

"Appendix A: Parameter Changes in Incompatible Macros" describes differences between the MVS/370 and MVS/XA parameters lists that downwardly incompatible macros pass to their service routines.

"Appendix B: Control Block Updates" lists control blocks that are new, updated, or deleted or that can reside anywhere in virtual storage (above or below 16 megabytes).

This *MVS/XA Conversion Notebook, Volume 1* does **not** describe:

- How to install the program products. The *Program Directory* shipped with the product describes the installation procedure.
- JES conversion information. The *MVS/XA Conversion Notebook, Volume 2*, however, contains charts that list which JES releases run with which releases of MVS/SP Version 2 Release 2.
- How to write programs that execute in 31-bit addressing mode. *SPL: 31-bit Addressing* contains that information. This *MVS/XA Conversion Notebook, Volume 1* does, however, describe Release 2.1.x system changes that take advantage of or support 31-bit addressing and the impact the changes have on user-written programs. For example, this volume of the *MVS/XA Conversion Notebook* lists control blocks that have been moved to the extended area of virtual storage in the various levels of MVS/SP Version 2 Release 1 and gives an example of how you can change programs to access them.

The phrase 'published external interfaces' refers to interfaces documented in the following publications:

- *OS/VS2 MVS JCL*, GC28-0692
- *OS/VS2 Supervisor Services and Macro Instructions*, GC28-1114
- *OS/VS2 TSO Command Language Reference*, GC28-0646
- *OS/VS2 Guide to Writing a Command Processor or Terminal Monitor Program*, GC28-0648
- *OS/VS2 Data Management Macro Instructions*, GC26-3873
- *OS/VS2 Access Method Services*, GC26-3841
- *OS/VS2 Virtual Storage Access Method (VSAM) Programmer's Guide*, GC26-3838
- *OS/VS2 MVS Data Management Service Guide*, GC26-3875
- *OS/VS2 System Programming Library: Data Management*, GC26-3830
- *MVS/370 Data Administration: Macro Instruction Reference*, GC26-4057
- *MVS/370 Integrated Catalog Administration: Access Method Services Reference*, GC26-4051
- *MVS/370 VSAM Catalog Administration: Access Method Services Reference*, GC26-4059
- *MVS/370 VSAM Administration Guide*, GC26-4066
- *MVS/XA Data Administration Guide*, GC26-4140
- *MVS/XA System-Data Administration*, GC26-4150
- *MVS/XA Data Administration: Macro Instruction Reference*, Version 1 GC26-4014
- *MVS/XA Data Administration: Macro Instruction Reference*, Version 2 GC26-4141
- *MVS/XA VSAM Administration Guide*, Version 1 GC26-4015
- *MVS/XA VSAM Administration Guide*, Version 2 GC26-4151
- *MVS/XA Integrated Catalog Administration: Access Method Services Reference*, Version 1 GC26-4019
- *MVS/XA Integrated Catalog Administration: Access Method Services Reference*, Version 2 GC26-4135
- *MVS/XA VSAM Catalog Administration: Access Method Services Reference*, Version 1 GC26-4075

- *MVS/XA VSAM Catalog Administration: Access Method Services Reference*, Version 2 GC26-4136
- *MVS/XA VSAM Administration: Macro Instruction Reference*, Version 1 GC26-4016
- *MVS/XA VSAM Administration: Macro Instruction Reference*, Version 2 GC26-4152

Submitting Conversion Hints

This book will be updated as more information becomes available. You can submit conversion hints for possible publication in this book. Use the reader's comment form or the conversion notebook input form at the back of this book or send your information to:

```
IBM Corporation
Publications Department
Department D58
PO Box 390
Poughkeepsie, New York 12602
ATTN:  MVS/Extended Architecture Conversion Notebook
```

It is understood that IBM and its affiliated companies shall have the nonexclusive right, in their discretion, to use, copy, and distribute all submitted information or material, in any form, for any and all purposes, without any obligation to the submitter.

When submitting conversion hints, please indicate from what system you are converting, the program products installed on it, and the program products being installed on the MVS/XA system.

Related Publications

The *MVS/XA Conversion Notebook, Volume 1* highlights differences in MVS/XA to help you identify changes you need to make to existing procedures, programs, and system modifications. You will, however, need other books in the MVS/XA library to make any required changes to your procedures and programs. The following table lists all the books referred to in the *MVS/XA Conversion Notebook, Volume 1*. The table shows the topic used in the reference, the corresponding full title, and the order number of the book. For a complete list of the publications that support MVS/XA, see the general information manuals.

Topic	Title	Order No.
commands	*MVS/Extended Architecture Operations: System Commands*	GC28-1206
conversion, JES3	*MVS/Extended Architecture JES3 Conversion Notebook*	SC28-1501
data management	*MVS/Extended Architecture System Programming Library: Data Management*	GC26-4010
devices	*Device Support Facilities User's Guide and Reference*	GC35-0033
DFP version 1	*MVS/Extended Architecture Data Facility Product Version 1: Planning Guide*	GC26-4040
DFP version 2	*MVS/Extended Architecture Data Facility Product Version 2: Planning Guide*	GC26-4147
diagnosis	*MVS/Extended Architecture Diagnostic Techniques*	LY28-1199
GIM for the BCP	*MVS/System Product Version 2 General Information Manual*	GC28-1118
GIM for DFDSS	*Data Facility Data Set Services: General Information*	GC26-1423
GIM for DFP Version 1	*MVS/Extended Architecture Data Facility Product Version 1: General Information*	GC26-4007
GIM for DFP Version 2	*MVS/Extended Architecture Data Facility Product Version 2: General Information*	GC26-4142
GIM for RMF	*Resource Measurement Facility General Information Manual*	GC28-1115
GIM for TSO/E	*TSO Extensions (TSO/E) General Information*	GC28-1061
initialization and tuning	*MVS/Extended Architecture System Programming Library: Initialization and Tuning*	GC28-1149
instruction set, System/370	*IBM System/370 Extended Architecture Principles of Operation*	SA22-7085
IOCP for the 308x	*Input/Output Configuration Program User's Guide and Reference (IBM 3081, 3083, 3084, 9081, 9083 Processor Complexes)*	GC28-1027
IOCP for the 3090	*Input/Output Configuration Program User's Guide and Reference (IBM 3090 Processor Complex)*	SC38-0038
IOCP for the 4381	*4381 Processor Input/Output Configuration Program User's Guide and Reference*	GC24-3964
IPCS	*MVS/Extended Architecture Interactive Problem Control System User's Guide and Reference*	GC28-1297
JES2	*MVS/Extended Architecture System Programming Library: JES2 Initialization and Tuning*	SC23-0065

Topic	Title	Order No.
JES2 (MVS/370)	*System Programming Library: JES2 User Modifications and Macros*	LC23-0067
JES2	*MVS/Extended Architecture System Programming Library: JES2 User Modification and Macros*	LC23-0069
JES3	*MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning*	SC23-0059
JES3	*MVS/Extended Architecture System Programming Library: JES3 User Modification and Macros*	LC28-1372
linkage editor and loader	*MVS/Extended Architecture Linkage Editor and Loader User's Guide, Version 1*	GC26-4011
macros	*MVS/Extended Architecture Supervisor Services and Macro Instructions*	GC28-1154
macros	*MVS/Extended Architecture System Programming Library: System Macros and Facilities Volumes 1 and 2*	GC28-1150 GC28-1151
messages	*MVS/Extended Architecture System Message Library: System Messages Vol. 1*	GC28-1376
messages	*MVS/Extended Architecture System Message Library: System Messages Vol. 2*	GC28-1377
MSS	*OS/VS Mass Storage Subsystem (MSS) Installation Planning and Table Create*	GC35-0028
MVSCP	*MVS/Extended Architecture: MVS Configuration Program Guide Guide and Reference*	GC28-1335
MVS/XA overview	*MVS/Extended Architecture Overview*	GC28-1348
operations	*IBM 3081 Operator's Guide for the System Console*	GC38-0034
operations	*IBM 3083 Operator's Guide for the System Console*	GC38-0036
operations	*IBM 3084 Operator's Guide for the System Console*	GC38-0037
operations	*IBM 4381 Processor Operations Manual*	GA24-3949
operations	*IBM 3090 Processor Complex Operator Controls for the System Console*	SC38-0040
operations	*IBM 3090 Model 200 Processor Complex Operator Tasks for the System Console*	SC38-0041
operations	*IBM 3090 Model 150 and 180 Processor Complex Operator Tasks for the System Console*	SC38-0049

Topic	Title	Order No.
operations	*IBM 3090 Model 400 Processor Complex Operator Tasks for the System Console*	SC38-0050
RACF	*Resource Access Control Facility (RACF) Command Language Reference*	SC28-0733
RACF	*Resource Access Control Facility (RACF) Security Administrator's Guide*	SC28-1340
RACF	*System Programming Library: Resource Access Control Facility*	SC28-1343
recovery and reconfiguration	*MVS/Extended Architecture Planning: Recovery and Reconfiguration*	GC28-1160
RMF	*MVS/Extended Architecture Resource Measurement Facility (RMF) Monitor I and Monitor II Reference and User's Guide*	LC28-1556
RMF	*MVS/Extended Architecture Resource Measurement Facility (RMF) Monitor III Reference and User's Guide*	LC28-1557
serialization	*OS/VS2 MVS Planning: Global Resource Planning*	GC28-1062
service aids	*MVS/Extended Architecture System Programming Library: Service Aids*	GC28-1159
SMF	*MVS/Extended Architecture System Programming Library: System Management Facilities*	GC28-1153
SMP/E	*System Modification Program Extended (SMP/E) User's Guide*	SC28-1302
system generation MVS/SP Version 1	*MVS/Extended Architecture Installation: System Generation Version 1*	GC26-4009
system generation MVS/SP Version 2 Release 1	*MVS/Extended Architecture Installation: System Generation*	GC26-4148
system modifications	*MVS/Extended Architecture System Programming Library: System Modifications*	GC28-1152
user exits	*MVS/Extended Architecture System Programming Library: User Exits*	GC28-1147
utilities	*MVS/Extended Architecture Data Administration: Utilities, Version 1*	GC26-4018
vector processing	*IBM System/370 Vector Operations*	SA22-7125
XRF, introduction	*MVS/Extended Architecture Introduction to Extended Recovery Facility (XRF)*	GC28-1135
XRF, planning	*MVS/Extended Architecture Planning: Extended Recovery Facility (XRF)*	GC28-1139
31-Bit addressing	*MVS/Extended Architecture System Programming Library: 31-Bit Addressing*	GC28-1158

Contents

Figures

Summary of Amendments

Summary of Amendments
for GC28-1143-6
MVS/Extended Architecture

Except for the topics listed above
of the information within
Conversion Notebook.
because information that has
has been removed from
new information, are
the MVS/SP Version

Topics with new information

- How this book is
- Handling Dev
- OPEN and
- OPEN and
- EOV and
- Checklist for
- Services Independent
- Limiting User

Topics that have been

- Conversion Tasks
- Providing a
- Creating or
- Coding MVS/370
- Rebuilding
- Changes to the
- Defining System
- Loading Programs
- Services with
- Performing I/O in
- Using the ASM
- Appendix B: Control

The topics that have been completely deleted are:

- Installing and MVS/XA System
- Deleting Display Control Modules (DCMs) from SYS1.LPALIB
- Resource Measurement Facility (RMF) Releases
- The JES2 Component of MVS/SP JES2 Version 2
- The JES3 Component of MVS/SP JES2 Version 2
- Devices Not Supported When Running MVS/XA

Summary of Amendments
for GC28-1143-5
MVS/Extended Architecture

This major revision contains conversion information for MVS/SP Version 2 Release 1.7. It also includes some technical changes not specifically related to this release as well as minor editorial changes. Bars (|) in the left-hand margin highlight the new information. Editorial changes are not barred.

Generally speaking, Release 1.7 includes support for the IBM 3090 model 400 processor complex and reconfiguration of expanded (also known as extended) storage. Release 1.7 also incorporates Release 1.3VFE and Release 1.3AE into the base control program.

With respect to conversion, Release 1.7 affects the following areas within MVS/XA:

- CONFIGxx member of SYS1.PARMLIB
- CONFIG command
- DISPLAY M command
- RMF Version 3 Release 4.1
- Rotate priority
- SMF records
- Updated control blocks

Summary of Amendments
for GC28-1143-4
MVS/Extended Architecture

This major revision contains conversion information for:

- MVS/SP Version 2 Release 1.3 Vector Facility Enhancement (Release 1.3VFE)

  This enhancement supports the Vector Facility available in IBM 3090 processor complexes. Users can make use of the Vector Facility by means of Vector FORTRAN and Assembler H Version 2.1.

- MVS/SP Version 2 Release 1.3 Availability Enhancement (Release 1.3AE)

  The availability enhancement represents changes in MVS/XA to support the extended recovery facility (XRF). Through enhancements to five software products, XRF improves the availability of interactive transaction processing to Information Management System/Virtual Storage (IMS/VS) end users. XRF provides an alternate processing environment that continually tracks the progress of the active environment and stands ready to carry on with current transaction processing if the active environment fails.

  In addition to MVS/SP Version 2 Release 1.3AE, the products that make up XRF are:

  - IMS/VS Version 2 Release 1
  - ACF/VTAM Version 3 for MVS/XA
  - ACF/Network Control Program Version 4
  - MVS/XA DFP Version 2 Release 1.0

- MVS/SP Version 2 Release 1.5

  Because this release represents changes to JES2 and JES3 components only, the only references to the release are charts in Chapter Two that show all the previous MVS/SP Version 2 releases that can run with the JES2 or JES3 shipped with Release 1.5.

  JES2 conversion information is available in:

  - *MVS/XA SPL: JES2 User Modifications and Macros* LC23-0069
  - *SPL: JES2 User Modifications and Macros* LC23-0067
  - *JES2 Program Directory*

  JES3 conversion information is available in:

  - *MVS/XA JES3 Conversion Notebook* SC28-1501
  - *JES3 Program Directory*

- MVS/XA DFP Version 2 Release 1.0

This revision also includes some technical changes not specifically related to these releases as well as minor editorial changes. Bars (|) in the left-hand margin highlight the new information. Editorial changes are not barred.

Note: Release 1.3VFE and Release 1.3AE are mutually exclusive releases of MVS/SP Version 2. This means you can install either one, but not both, in a processor complex running in a single-system image or on one side of a partitioned processor complex. Similarly, information in this book specifically related to one enhancement does not apply to the other. Be sure to note the paragraph and chart headings to determine which enhancement is being described.

The areas in MVS/XA and related products affected by the releases documented in this edition are:

- Assembler H Version 2.1
- AVM member of SYS1.PROCLIB
- CALLDISP macro

- Catalog Address Space
- Dumps (SNAP, ABDUMP, and stand-alone)
- Linkage editor
- PRDMP control statement, AVMDATA
- Program fetch
- Program properties table (PPT)
- RMF Version 3
- SADMP
- SMF records
- The CONFIG, DISPLAY, MODE and START commands
- Several new and updated control blocks

The additional technical changes in the book affect the following topics:

- Data Facility Data Set Services (DFDSS)
- IPCS and PRDMP
- The JCL REGION parameter
- [illegible] GENERATE function
- [illegible] macro
- [illegible]
- [illegible] by requests for price quotations (RPQ)s

Summary of Amendments

[illegible]

MVS/Extended Architecture

This major revision contains conversion information for Release 1.3 of MVS/SP and [illegible] MVS/XA DFP Release 1.2. It also includes some technical changes not specifically related to this release as well as minor editorial changes. [illegible] the left-hand margin highlight the new information. Editorial changes are not marked.

Generally speaking, Release 1.3 updates support the IBM 3090 model 200 processor complex and expanded (also known as extended) storage. For conversion purposes, this release affects the following areas within MVS/XA:

- [illegible] instruction
- DISPLAY M command
- [illegible] macro
- [illegible]
- [illegible] member of SYS1.PARMLIB
- PRDMP control statement, SADMPMSG
- RMF reports
- SMF records
- SRM control blocks
- Several new and updated control blocks

The additional technical changes in the book affect the following topics:

- Resource Access Control Facility (RACF)
- TSO/Extensions (TSO/E)

Chapter 1. Introduction

Conversion to MVS/Extended Architecture (MVS/XA) is the process of installing the program products that will comprise your MVS/XA system, making any required changes to existing programs and procedures, and running and testing the new system as the production system.

The work required to convert to MVS/XA varies greatly from one installation to another and depends on:

- The level of the MVS/370 system to be converted. The more your current system resembles your target system, the less work you have to do at the same time you install the MVS/XA components. The next topic describes several ways you can prepare your MVS/370 system for conversion to MVS/XA.

- The number of programs that must be modified. Early installers reported that none of their high-level language programs had to be changed. About fifteen percent of their authorized assembler language programs required modification.

  With few exceptions, user-written assembler language programs that use only unauthorized services and published external interfaces will run unchanged. Many programs that use authorized services or undocumented interfaces will also work unchanged, but you might have to modify some. Specifically, you need to modify programs that depend on the structure and content of system control blocks or interfaces that are changed. The changed interfaces are almost exclusively authorized, internal interfaces.

- The number and type of modifications your installation has made to MVS/370 that must be adapted to MVS/XA, and which components your installation has modified. Some components are changed more than others.

In general, for the BCP, there is a high degree of compatibility between MVS/370 and MVS/XA:

- Exit interfaces, in general, are unchanged or compatibly expanded.

- You do not have to recompile or relink edit existing application programs, unless you change them.

- MVS/370 JCL and JES control statements will work in MVS/XA. In some instances, however, you might have to change JCL specifications, including:
  - DD statements for SYS1.DUMPnn data sets. The DD statements must specify DISP=SHR.
  - The REGION parameter on a linkage editor job.
  - JCL that specifies programs not supported in MVS/XA (for example, IEHDASDR).
  - JCL that specifies unsupported devices.
- MVS/XA uses the same system data sets as MVS/370. Changes have been made to SYS1.PARMLIB members.

Conversion Tasks You Can Do Before Installing MVS/XA

You can stage the conversion to MVS/XA by performing many of the conversion tasks on your MVS/370 system before installing the MVS/XA components. Moving in the direction of MVS/XA as early as possible has several advantages. The most obvious is that it minimizes the activities you must perform at the same time you install MVS/SP Version 2 and MVS/XA DFP. In addition, you become familiar with the new environment gradually and have less to learn all at once. Finally, the MVS/370 system will be in the best position for coexistence and back-up. MVS/370 and MVS/XA can operate and share data in the same installation.

To prepare for MVS/XA, you can:

- Upgrade your system to at least MVS/SP - JES3 Version 1 Release 3.4 or MVS/SP - JES2 Version 1 Release 3.4. If the JES component of your MVS/370 system is already at one of these levels, you may not have to install the JES component included with Release 2.2.x. Refer to the *MVS/XA Conversion Notebook, Volume 2*, for an explanation of which JES releases work with the various levels of MVS/SP Version 2 Release 2.
- Install the MVS/XA-compatible levels of other program products that your installation needs in MVS/XA. The IBM Announcement letter (285-347), "Programs Supported in a Multiple Virtual Storage/ Extended Architecture (MVS/XA) Environment," lists the program products, IBM Field Developed Programs (FDPs), Installed User Programs (IUPs), and Program Offerings that can be installed on and will operate with both MVS/370 and MVS/XA. *Volume 2* lists program products specifically required for Release 2.2.x.

  When initially installing an MVS/XA-compatible program product on MVS/370, have your IBM representative check the RETAIN Preventive Service Planning (PSP) bucket for that product. Some products might require PTFs to ensure compatibility with MVS/XA. Have RETAIN checked again just before testing the product under MVS/XA. *Volume 2* lists program products which must have PTFs in order to take advantage of all of the Release 2.2.x enhancements and updates.

- Install products whose functions are included in MVS/XA such as MVS/370 Data Facility Product (MVS/370 DFP).
- Review the devices and functions that are not supported in MVS/XA. If you are currently using any of them, migrate to the successor product or function.
- Check the RPQ devices or features you have on your system to determine if they will work in MVS/XA. *Volume 2* lists devices not supported by Release 2.2.x along with devices for which RPQs have been approved.
- Install compatibility PTFs on your MVS/370 system and reassemble the affected programs. See the following topics in Chapter 3:
  - "IOHALT Macro Instruction (SVC 33)" on page 3-9
  - "IOSGEN UCBLOOK Macro Instruction" on page 3-9
- Identify and make required programming changes that can be made on your MVS/370 system.

Publications Changes

For the most part, the MVS/XA publications are technically updated versions of their MVS/370 counterparts, reissued with new order numbers. The title pages of most MVS/SP Version 2 and MVS/XA DFP publications include "MVS/Extended Architecture" to allow you to easily distinguish between MVS/370 and MVS/XA publications. The MVS/XA library also adds new books, deletes others, and reorganizes some books. All component-level diagnostic techniques, for example, move from the *MVS/XA Diagnostic Techniques* manual to appropriate volumes of the *System Logic Library*

Chapter 2. Installation and Initialization

This chapter contains information related to installing an MVS/XA system, initializing it, and generating the stand-alone dump program. Topics related to installing MVS/XA are:

Topics related to initializing MVS/XA are:

Topics related to generating stand-alone dump include:

System Generation

You can generate an MVS/XA system on either:

- An MVS/370 system that is at least at the OS/VS2 Release 3.8 level. The MVS/370 system must also support the device types on which the MVS/XA system libraries are to reside.

- An MVS/XA system.

Required Environment for Generating MVS/XA

The following program products must be installed on the system used to build the MVS/XA system:

- Assembler H Version 2
- The linkage editor in MVS/XA DFP or MVS/370 DFP
- SMP Release 4 or SMP/E
- Device Support Facilities Release 6 or a subsequent release

  The Device Support Facilities product is required to write the IPL text and to initialize the DASD volumes on which the new system will reside.

In addition, DFDSS (Data Facility Data Set Services) or an equivalent dump/restore product is recommended to make a backup copy of the new system. IEHDASDR does not work in MVS/XA. Furthermore, DFDSS cannot restore data dumped using IEHDASDR.

Beginning with Release 2.1.2, DFDSS runs on both MVS/370 and MVS/XA. Using DFDSS you can dump data on one system and restore it on the other.

Providing a Backup Copy of the Existing System

Before using the SMP ACCEPT function to incorporate the MVS/XA products into your DLIBs, copy the DLIBs using DFDSS or an equivalent product. MVS/SP Version 2 completely replaces (and, therefore, deletes from the existing DLIBs), the base control program (BCP) in MVS Release 3.8. MVS/XA DFP completely replaces all MVS/370 modules containing the functions that MVS/XA DFP provides. The ACCEPT function also deletes:

- System Activity Measurement Facility (MF/1)
- Display Exception Monitor Facility (DEMF)
- The External Writer
- TSO TEST
- The TSO command package. The functions in that package are:
  - Support for running terminal sessions as batch jobs
  - Automatic saving of data
  - Accounting facilities enhancements
  - Defaults for the user attribute data set
  - ATTRIB and FREE subcommands
  - ALL keyword for the FREE command and subcommand
  - Eight-character station ID

- TSO/E for MVS/370, which includes the TSO command package
- All service information for the deleted modules

MF/1 and DEMF are not replaced. With MVS/XA, the MF/1 functions are provided by RMF Version 3 and the DEMF functions are provided by NPDA Version 2 or subsequent releases. TSO TEST, the TSO command package, and the TSO/E functions are included in TSO/E with the MVS/XA feature (5665-285). If your installation requires these functions, install TSO/E with the MVS/XA feature. The External Writer function is incorporated into the MVS/SP Version 2 BCP. The program directories for MVS/XA DFP and MVS/SP Version 2 list the FMIDs that MVS/XA replaces.

Once the DLIBs are updated, there is no simple way to restore them to the MVS/370 level unless you have a backup copy. The SMP RESTORE function cannot restore DLIBs.

Defining Devices

MVS/XA supports a maximum of 4096 devices. However, the number that your installation can actually connect depends on the processor model and is given in the *IOCP User's Guide and Reference* for your type of processor complex. The *MVSCP User's Guide and Reference* explains how to define devices to the MVS/XA system.

MVS/370 allows no more than 1917 devices because UCB pointers are only 2-bytes long. MVS/XA removes that limitation by using 3-byte UCB pointers.

When to Use the I/O Configuration Program (IOCP)

You need to use IOCP to create a new I/O configuration data set (IOCDS) in four situations:

- Whenever you are migrating from MVS/370 to MVS/XA.
- When you are altering the I/O configuration in the processor complex.

  (For example, when you are adding I/O devices, adding channel paths, or moving an I/O device from one channel path to another.)
- When you are migrating to a processor complex with a different type of channel subsystem.
- When a new level of IOCP for your processor becomes available.

Migrating from one release of MVS/XA to another does not always require a new IOCDS.

Each processor complex comes with the stand-alone version of IOCP that you can use to define the I/O configuration before IPL. For some processor complexes, there is also a version of IOCP that runs under MVS.

More than one MVS version of IOCP may be packaged with a particular MVS/SP Version 2 release. Figure 2-1 shows which MVS versions of IOCP are compatible with the various IBM processor complexes. The chart also lists the publications that explain how to use each MVS version of IOCP.

IBM Processor Complex	MVS IOCP Program	Publications
4381	none	*4381 IOCP User's Guide and Reference* GC24-3964
308x, 908x	ICPIOCP	*IOCP User's Guide and Reference* GC28-1027
3090, 9190	IOPIOCP	*IOCP User's Guide and Reference* SC38-0038

Figure 2-1. IBM Processor Complexes and Versions of IOCP

Even though different MVS versions of IOCP programs exist, you can use either one on any processor for testing your IOCP input data. However, regardless of where it runs, you must use the MVS version of IOCP compatible with your target processor complex to produce an IOCDS compatible with your target processor complex. Furthermore, the MVS version of IOCP compatible with your target processor complex must be running on the target processor complex when it writes the IOCDS you intend to use.

Note: The *IOCP User's Guide and Reference* for the IBM 3090 processor complex includes a preface with introductory information for installations migrating from a 308x processor complex to an IBM 3090 processor complex.

Creating or Modifying an IOCDS Using the MVS Version of IOCP

The procedure for creating or modifying an IOCDS depends upon the type of processor that will run the newly installed release of MVS/XA. The *IOCP User's Guide and Reference* for the target processor explains the appropriate procedure for creating a new IOCDS.

Installations running an IBM 3090 model 400 processor complex can use the model 400 IOCP to generate an IOCDS on a model 200 . However, to do so you must first apply MEC # 225615 to your IBM 3090 model 200 processor complex.

Processor Complexes That Have System/370 and 370-XA Modes

During the migration period, it is important to create an IOCDS that can be used in both System/370 and 370-XA modes. Therefore, you must run the 370/370-XA IOCP to create the IOCDS. In addition, to ensure that the IOCDS works for both System/370 and 370-XA, use the same IOCP macro specifications to create it as you use to create a System/370 IOCDS. However, be aware that although the macro instructions are upward compatible between System/370 and 370-XA modes, there are differences in the way IOCP treats the macro specifications when defining a 370-XA I/O configuration. Most of the differences support the new I/O architecture:

Macro	MVS/XA Differences
CHPID	IOCP requires channel numbers and channel sets only for devices to be used in System/370 mode. The 370-XA architecture does not use channel numbers and channel sets.
IODEVICE	**ADDRESS keyword.** IOCP treats the ADDRESS keyword value as a device address in System/370 mode and as a device number in 370-XA mode. During the conversion period, specify the device numbers exactly the way you specify System/370 device addresses. From a users point of view, MVS/XA device numbers are equivalent to MVS/370 device addresses (sometimes referred to as CUAs in MVS/370). Both uniquely identify a device. In some publications and messages, you might still see device numbers referred to as device addresses. **UNITADD keyword.** UNITADD is a an optional keyword that has meaning only in 370-XA mode. It specifies the two-digit physical unit address of the device being described. UNITADD provides an alternative to specifying the unit address on the ADDRESS keyword (the last two digits of ADDRESS=xxx). If you specify UNITADD, the last two digits of ADDRESS=xxx need not be the device's actual physical unit address, as previously required. Instead, they can be any value that: (a) makes the device number unique, and (b) follows the rules listed in the *IOCP User's Guide and Reference* for your processor complex. UNITADD allows you to assign the same unit address to as many devices as the number of channel paths. Without UNITADD, the limit is sixteen because the first digit of ADDRESS=xxx must be 0-F, the last two digits must be the device's unit address, and the three-digit combination must be unique. The first two restrictions allow only sixteen unique combinations (for example, 0FF-FFF for devices having unit address FF). You cannot use UNITADD on IODEVICE macros used to generate an MVS/370 system. The MVS/370 SYSGEN program does not recognize UNITADD on the IODEVICE macro, and fails. **PATH keyword.** PATH is an optional keyword that has meaning only in 370-XA mode. It specifies a preferred path, which the channel subsystem tries first when initiating I/O to the device. You cannot include PATH on IODEVICE macros used to generate an MVS/370 system. As with UNITADD, the MVS/370 SYSGEN program does not recognize the PATH keyword on the IODEVICE macro, and fails. **STADET keyword.** New for Release 2.1.3, this optional keyword can be used in both System/370 and 370-XA modes. However, it only has meaning in 370-XA mode in an IBM 3090 processor complex. STADET provides input for IOPIOCP. **Note.** As of Release 2.1.7, regardless of how they are actually written, the system interprets all uses of the STADET keyword as STADET=N. Contact the Systems Engineer in your IBM branch office for details on the STADET keyword.

WARNING: When coding the CNTLUNIT macro, remember to specify on the UNITADD parameter **all** unit addresses that the control unit can address, regardless of whether a device is actually attached. This rule is not new.

Review your current I/O configuration to ensure that it is valid in 370-XA mode, **if you ever plan to run only in System/370 mode.** The 370/370-XA IOCP imposes restrictions on the I/O configuration (for example, no physical control unit can share devices with more than three other physical control units). Current I/O configurations might violate the restrictions, in which case the 370/370-XA IOCP issues error messages. The *IOCP User's Guide and Reference* for your type of processor complex describes the I/O configuration requirements you must satisfy.

Coding Macros Used for SYSGEN with MVS/370

Many of the MVS/370 SYSGEN macros are obsolete as of Release 2.2.0; others are used as input to the MVS configuration program (MVSCP), and four remain as input to system generation: DATAMGT, DATASET, GENERATE and JES.

Figure 2-2 summarizes the changes to SYSGEN macros that occurred during Release 2.1.x. The *MVS/XA Conversion Notebook, Volume 2* summarizes the Release 2.2.x changes to system generation macros.

Most of the changes are compatible. The system generation process generally ignores MVS/370 macros, keywords, and options that have no meaning. In some cases, it accepts them and issues an informational or warning message. However, you do need to check for macros that specify unsupported device types. (See the *Volume 2* for a list of such devices.) The macros you need to check include DATASET, GENERATE, and IODEVICE.

Macro	Description of Updates
CHANNEL	Obsolete. If SYSGEN processing encounters a CHANNEL macro, it issues an informational message and continues.
CTRLPROG	The CHR keywords is obsolete as are the following keywords and options. However, when these are encountered the system produces an explanatory MNOTE: - ACRCODE - STORAGE - WARN
DATASET	A new DUMPDSN keyword specifies the suffixes or ranges of suffixes for dump data set names. You can define up to 100 dump data sets. Earlier releases allow a maximum of 10. You can specify secondary extents for the PARMLIB data set.
GENERATE	The default name for the INDEX keyword is SYSX, instead of SYS1.
IODEVICE	The AP and OPTCHAN keywords are ignored. GCU is obsolete. The 2-CHANSW option on the FEATURE keyword is ignored. The following options are obsolete: - ABSLTVEC - BUFFER4K - BUFFER8K - CHARGNTR - DATACONV - DEKYB2260 - DESIGNFEAT - LIGHTPEN - LINEADDR - MDECOMPAT - NMKEYB2260 - NODESCUR - READWRITE
SCHEDULR	TAVR=200 is obsolete.
UNITNAME	The maximum number of unique groups allowed is increased from 100 to 256. The maximum number of device numbers allowed is 4112 minus the number of unique groups. Earlier releases allow 2056. You can include a maximum of 4111 device numbers in one group.

Figure 2-2. Obsolete/Updated SYSGEN Macros, Keywords, and Options

Rebuilding Alternate Eligible Device Tables (EDTs)

EDTs are not compatible between MVS/370 and MVS/XA. Neither an MVS/XA nor an MVS/370 system can use EDTs verified on the other system. If your installation uses alternate EDTs, you must rebuild them using the MVS configuration program (MVSCP). You can build the EDTs on either an MVS/370 or an MVS/XA system, but you must verify them on an MVS/XA system.

Changes to the Program Properties Table (PPT)

Listed below are two situations that require a new PPT. When one occurs, if your installation has added entries to the existing PPT, either copy the installation entries from the existing PPT into the new PPT or copy the entries in the new PPT into the existing PPT. *MVS/XA SPL: Initialization and Tuning* describes how to update the PPT. *MVS/XA SPL: System Modifications* has details on the PPT format.

1. Installing Release 2.1.1

   With Release 2.1.1, the updated PPT contains two new entries: one for IFASMF and one for CSVLLCRE. IFASMF is a new SMF module that is required to start the new SMF address space. CSVLLCRE creates and maintains a new directory of modules in the LNKLST concatenation. (See "Using a New Directory for LNKLST Data Sets" for more information.)

2. Installing Release 2.1.3AE

   Release 2.1.3AE adds the load module, AVFMNBLD, that starts the availability manager address space.

The Vector Facility Enhancement and the PPT

Installations with processor complexes that include the Vector Facility should not specify processor affinity for programs in the PPT. MVS/XA dynamically manages processor affinity to insure that the programs using vector instructions run on a processor with the Vector Facility when necessary. Setting processor affinity in the PPT may interfere with this process.

Defining System Data Sets

An MVS/XA system requires data sets with the same names and characteristics as an MVS/370 system. Additional information related to defining system data sets is described below.

Device Types Allowed

Except for page data sets, you can place system data sets on all devices that were previously allowed, provided MVS/XA supports those device types. The next paragraph describes the page data set exception. You must move data sets on unsupported devices to supported devices. See the *MVS/XA Conversion Notebook, Volume 2* for a list of such devices.

Swap Data Sets

You need to evaluate the number and size of swap data sets defined. As virtual storage requirements increase, you might need to define additional swap space.

Dump Data Sets

Installations can now define up to 100 SYS1.DUMPnn data sets. A maximum of 10 dump data sets are allowed in MVS/370.

Your installation might want to increase the number and size of dump data sets defined during the migration period. Allocate dump data sets large enough to contain the maximum size SVC dump expected.

The size of the dump depends on the dump options and the size of your application programs. If these programs are the same size on your MVS/370 and MVS/XA systems, your MVS/370 SYS1.DUMPnn data sets may be large enough for the MVS/XA system. Once your MVS/XA system is running you can, if you need to, reallocate SYS1.DUMPnn data sets so that there is a closer match between them and the size of the SVC dumps that occur.

A new command, DUMPDS, allows installations to add and delete SYS1.DUMPnn data sets after IPL/NIP time. See "Summary of New, Updated, or Deleted Commands" in Chapter 4. Also, SYS1.DUMPnn data sets must be allocated DISP=SHR instead of DISP=OLD. See "JCL Changes to Jobs that Allocate SYS1.DUMP Data Sets" in Chapter 4.

SYS1.DAE Data Set

Beginning with Release 2.1.1, you must allocate a system data set, SYS1.DAE, at IPL time in order to use dump analysis and elimination (DAE). DAE stores symptom information from dumps it identifies as unique in SYS1.DAE and uses that information when determining if subsequent dumps are duplicates. "Dump Analysis and Elimination (DAE)" in Chapter 6 gives an overview of DAE and describes in more detail how and when SYS1.DAE is used.

You can create SYS1.DAE using JCL in the DAEALLOC member of SYS1.SAMPLIB. (The DATASET system generation macro does not support SYS1.DAE.) For instructions, see *MVS/XA SPL: System Modifications.*

MSTJCLxx Members in the SYS1.LINKLIB Data Set

As of Release 2.1.1, the JCL for starting the master scheduler address space is contained in MSTJCLxx members of SYS1.LINKLIB. MSTRJCL, the SYS1.LINKLIB member that earlier releases use, is deleted. Instead, IBM supplies default JCL in member MSTJCL00.

Concatenating Data Sets to the SYS1.LPALIB Data Set

Beginning with Release 2.1.1, you can concatenate data sets to the SYS1.LPALIB data set. The system uses the modules in the concatenated data sets, as well as the SYS1.LPALIB data set, to build the PLPA, the MLPA, and the FLPA. Earlier releases of MVS use only the modules in the SYS1.LPALIB. SYS1.LPALIB concatenation allows you to share a single SYS1.LPALIB data set

among several systems, yet still tailor the PLPA, MLPA, and FLPA of each system by varying the concatenation.

To concatenate data sets:

- List in the LPALSTxx PARMLIB member which data sets are to be concatenated. The data sets must be included in the master catalog and must be APF authorized.

- Specify on the LPA system parameter which LPALSTxx members are to be processed. You can include the LPA parameter in IEASYSxx, or an operator can specify it when prompted for system parameters. If you omit the LPA parameter, the system uses SYS1.LPALIB only, as in previous releases of MVS.

See *MVS/XA SPL: Initialization and Tuning* for more detail on creating LPALSTxx members and specifying the LPA system parameter.

SYS1.LOGREC Data Sets

Beginning with Release 2.1.1, you can place SYS1.LOGREC data sets on a volume other than the SYSRES volume. Several systems can then share a SYSRES volume and still have separate SYS1.LOGREC data sets. To use an alternate SYS1.LOGREC data set, simply include a data set named SYS1.LOGREC in the master catalog. The system searches for a data set of that name first in the master catalog, then in the SYSRES volume.

You might want to increase the size of your SYS1.LOGREC data set because MVS/XA can produce more diagnostic information.

SYS1.NUCLEUS Data Set

As in MVS/370, the MVS/XA SYS1.NUCLEUS must be a single extent. If you attempt to allocate a multiple extent data set, MVS/XA enters a restartable wait state (wait state code x'081'). MVS/370 takes different actions.

SYS1.PARMLIB Data Set

In MVS/XA, the SYS1.PARMLIB data set can be blocked and can have multiple extents. In MVS/370, the PARMLIB has to be unblocked and a single extent. Also see "New, Updated, or Deleted PARMLIB Members" on page 2-19.

System Data Set Qualifiers

You cannot specify a system data set qualifier of SYS1 on the INDEX parameter of the GENERATE macro. You can either specify some other high level qualifier or let system generation processing assign the default (SYSX). Stage II system generation processing changes the high level qualifiers to SYS1. This restriction is not new. It always applies when using a system other than a starter system to do a complete system generation. Note that the default high level qualifier is changed from SYS1 to SYSX.

Initializing DASD

You must use Device Support Facilities to initialize DASD volumes. IEHDASDR is no longer supported. See the *Device Support Facilities User's Guide and Reference* for directions.

Activating the Resource Access Control Facility (RACF)

If you intend to install RACF on your new system you should activate it (even before any data sets are protected) before testing your new system. MVS/XA systems with Release 2.1.2 or later (and MVS/370 systems with MVS/370 DFP 1.1 or later) support the always-call function. This function invokes the system authorization facility (SAF) which, in turn, invokes RACF each time a request to access a data set occurs.

If RACF is installed but not active, it processes authorization checking in RACF failsoft mode which prompts the operator for permission to access each data set. These queries can make testing new systems intolerably slow. Refer to the *SPL: RACF* for details on RACF failsoft processing.

Loading Programs

To IPL MVS/XA, you must use the IPL text distributed with MVS/SP Version 2 and follow the directions in the *Program Directory* for the release. The MVS/370 and MVS/XA IPL programs are not compatible. You must use Release 6 or a subsequent release of the Device Support Facilities to write the MVS/XA IPL text to DASD.

The Initial Program Load process loads the second MVS/XA bootstrap record into the frame at main storage absolute address 8 K. It loads the IPL text into main storage frame 0.

If you write your own bootstrap programs, (for example, to invoke SADMP) make sure that the addresses used to load the bootstrap records are in storage that will not be taken offline.

For MVS/XA, running on IBM 308x or IBM 3090 processor complexes that can be partitioned your choices are:

- Main storage frame 0, 2, 4, 6, and so on, throughout the first 2 megabytes of main storage.
- The low end of the highest storage range specified on the CONFIG frame on the system console. That storage range always remains online.

For MVS/XA running non-partitionable IBM 3090s and IBM 4381s, load your bootstrap programs beginning with main storage frame 0 and continuing through any frames in the first megabyte of main storage.

Loading the Microcode EC Tapes for Mass Storage Subsystems

To load MSS microcode in an MVS/XA environment, use the **MSC Table Create** (MSCTC) utility with PTF UZ09020 installed, instead of **IEHDASDR.** IEHDASDR does not work in MVS/XA.

The MSCTC control statement to use is **CREATE, the required parameter is** RESTOREC. For more information, see ***MSS Installation Planning and Table Create.***

System Parameter and SYS1.PARMLIB Considerations

The topics in this section contain information related to specifying system parameters. "New, Updated, or Deleted PARMLIB Members" on page 2-19 summarizes Release 2.1.x changes to SYS1.PARMLIB members.

Many parameters affect virtual storage address space areas. *MVS/XA SPL: Initialization and Tuning* and the *MVS/XA Overview* give background information on each portion of an MVS/XA address space.

Fixed Storage for SLIP Command Processors (IEASLPxx)

When the system processes SLIP commands at IPL time, it allocates fixed storage for the SLIP action processors and the control blocks they use. If your installation does not use SLIP commands in MVS/370, the fixed storage requirement is new. If your installation does use SLIP commands in MVS/370, because the fixed storage is allocated in virtual storage above 16 megabytes, you may gain approximately 31 K bytes of real storage below 16 megabytes.

Specifying the Reconfigurable Storage Unit (RSU) Parameter (IEASYSxx)

If you specified an RSU parameter in the past, when initializing Release 2.1.1 you need to review that specification. Release 2.1.1 satisfies the RSU request in a different way than previous releases do. The same RSU value might result in less reconfigurable storage.

Beginning with Release 2.1.1, the RSU parameter specifies *the total number of storage units MVS is to mark reconfigurable.* The system attempts to satisfy the request using offline storage units. It marks online storage reconfigurable only if there is not enough offline storage. After satisfying the RSU request, the system marks all remaining storage units (both online and offline) as preferred. If the system cannot satisfy the RSU request, the operator receives message IAR004I, as in previous releases.

In Release 2.1.0 and in MVS/370, the RSU parameter specifies the *number of online storage units to be marked reconfigurable when initializing the system.* Those releases use only online storage to satisfy the request. However, they also automatically mark storage that is offline at IPL time as reconfigurable when bringing it online. Thus, the total amount of reconfigurable storage is the amount marked reconfigurable when satisfying the RSU parameter, plus the amount brought online after the IPL. As a result, the RSU parameter in these releases can be less than the resulting amount of reconfigurable storage.

For processor complexes that can be partitioned to support independent operating systems, or that support removable storage elements, the RSU value needs to be equivalent to at least the amount of storage you plan to take offline before the next IPL. Some installations specify one additional storage unit to increase the probability that storage can be taken offline later. Remember that you can lose reconfigurable storage during normal processing in two ways:

- If the system runs out of preferred storage frames, it dynamically converts some reconfigurable storage to preferred storage.

- The system might not be able to reconfigure storage that contains storage errors that cannot be corrected.

Do not, however, specify more reconfigurable storage than you anticipate needing. Specifying too much can negatively affect performance.

If performing an IPL on a processor complex that cannot be partitioned to support independent operating systems, or does not support removal of storage elements, you do not need to specify an RSU value. The default RSU value of zero takes effect.

MVS/XA Recovery and Reconfiguration and *MVS/XA SPL: Initialization and Tuning* contain more detail on the RSU parameter.

Increasing the Minimum SQA Allocation (IEASYSxx)

If in MVS/370 you changed the NVTNVSQA field in module IEAVNIP0 to increase the minimum SQA allocation during previous system initializations, you need to read this topic. During system initialization, if the PAGE parameter specifies a large number of page data sets or if several 2305 Model 2 page data sets are active, the system's minimum allocation for SQA and extended SQA (seven 64 K blocks) might be depleted before the system processes the SQA parameter. In MVS/370, you can solve that problem by changing the contents of the NVTNVSQA field. That method does not work in MVS/XA. You can, however, increase the minimum allocations by changing the halfwords NVSQA and/or NVESQA in module IEAIPL04. Consult microfiche for the locations of these fields. If you increase the minimum SQA and/or extended SQA allocations and you want the total SQA size to remain the same, decrease the corresponding value on the SQA parameter.

Beginning with Release 2.1.2, more SQA and extended SQA are available earlier in the initialization process. As a result, you might not need to change the minimum SQA allocation:

- The minimum SQA allocation is increased to four 64 K blocks. In earlier releases, it is three.

- The system processes the SQA parameter earlier during system initialization.

Specifying the Size of Extended CSA and Extended SQA

You can use the CSA and SQA parameters in the IEASYSxx member of SYS1.PARMLIB to specify the size of CSA and SQA in virtual storage below, but not above, 16 megabytes. MVS/XA assigns default sizes for extended CSA and extended SQA (CSA and SQA in virtual storage above 16 megabytes). To override those defaults, use new options on the CSA and SQA system parameters.

The default CSA sizes are:

CSA -	100 K
extended CSA -	100 K

The CSA parameter has an additional option for specifying the size of extended CSA:

CSA = (a,b) where "a" specifies the size of CSA
and "b" specifies the size of extended CSA

The CSA values indicate the number of 1 K units to be reserved. The values override the default specifications. For example:

```
CSA = (100,200)  results in:

Reserved CSA = 100 K
Reserved extended CSA = 200 K
```

The default SQA sizes are:

SQA - 256 K
extended SQA - 256 K plus approximately 8 megabytes

The SQA parameter has an additional option for specifying the size of extended SQA:

SQA = (a,b) where "a" specifies the size of SQA
and "b" specifies the size of extended SQA

The SQA values specify the number of 64 K blocks MVS/XA is to reserve **in addition to** the minimum amount of storage it allocates for SQA and extended SQA. The minimum amounts are 256 K for SQA and 256 K plus approximately 8 megabytes for extended SQA. For example:

```
SQA = (3,5) results in:

Reserved SQA = 3 x 64 K + 256 K
Reserved extended SQA = 5 x 64 K + 256 K + approximately
                        8 megabytes
```

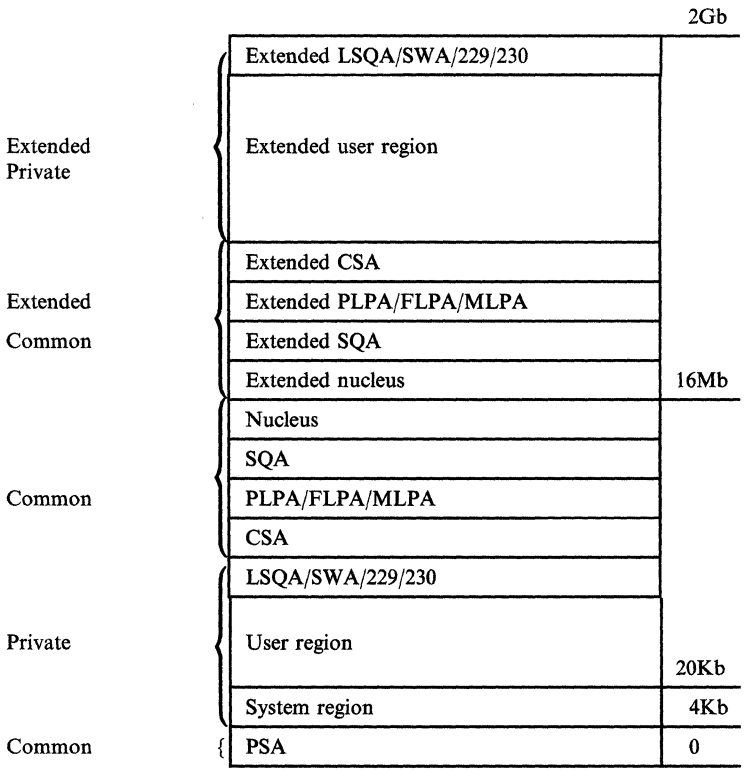
The default SQA values are SQA=(1,0).

If your system includes the Resource Measurement Facility (RMF), you can use the virtual storage report provided to compare your settings for CSA and SQA with the amount of these storage areas actually in use.

See *MVS/XA SPL: Initialization and Tuning* for more information on virtual storage areas.

Minimizing Private Area Storage Lost Because of Rounding (IEASYSxx)

Because the segment size in MVS/XA increases from 64 K to 1 megabyte, you need to pay closer attention to the amount of private area storage below 16 megabytes potentially lost to CSA at initialization time because of rounding. During IPL processing, MVS/XA builds the common area below 16 megabytes beginning at the 16 megabyte address and working down.



To determine the lower boundary of the common area (which is the upper boundary of the private area), MVS/XA rounds the bottom CSA address to the the next lowest megabyte. Thus, as much as 1020 K bytes (1 megabyte minus 4 K) of virtual storage can be added to the CSA, and consequently lost from the private area, because of rounding. Although it is not expected that your installation will lose that much private area, choose your CSA and SQA parameters carefully. If the size of the private area falls below 8 megabytes, MVS/XA informs the operator. You can keep track of rounding with the program product, Resource Measurement Facility (RMF). The virtual storage report provides a storage map showing the boundaries of the major system areas.

MVS/XA also builds a common area above 16 megabytes. Private area storage above 16 megabytes can also be lost because of rounding. However, users are not expected to have virtual storage constraint problems above 16 megabytes.

Specifying Dump Data Sets (IEASYSxx)

An additional operand of the DUMP parameter in IEASYSxx, 'DASD,xx-yy', allows an installation to request that the system being initialized use a specific range of DASD dump data sets. The DASD operand was redesigned specifically for systems that use SYS1.DUMP data sets that can be accessed by other systems. An installation can specify unique dump data sets for each system, which prevents SDUMP routines from using a dump data set concurrently for different systems.

The DASD operand also shortens dump data set catalog processing at initialization time. When specific dump data sets are indicated, the initialization routines do not have to issue all 100 locates to determine which dump data sets are cataloged.

Requesting Storage for RMF I/O Measurements (IEASYSxx)

If your installation wants RMF I/O data for device classes other than tape or DASD, you must request storage at initialization time for the control blocks in which the data is to be collected. SRM collects the I/O data that RMF uses in new control blocks, one per device. To request control block storage, specify on the new CMB parameter in IEASYSxx the non-tape and non-DASD device classes for which I/O data is to be collected.

Controlling the Number of Available ASVT Entries (IEASYSxx)

A system with Release 2.1.2 installed, creates and manages the address space vector table (ASVT) differently. The changes are designed to prevent the system from running out of ASVT entries.

When creating the ASVT, the system adds extra entries and reserves them for use when no unreserved entries are available. It uses one group of reserved entries only for address spaces being created in response to a START command. It uses a second group as replacements for entries that are not reusable because of latent cross-memory binds.

Two new system parameters allow your installation to specify the number of entries to be reserved for each purpose:

RSVSTRT Specifies the number of entries to be reserved for address spaces created in response to a START command. The default is five.

RSVNONR Specifies the number of entries to be used as replacements for entries that are not reusable. The default is also five.

The system still uses the MAXUSER parameter to limit the number of jobs and STARTed tasks that can execute concurrently under normal conditions. However, MAXUSER no longer specifies the maximum number of jobs or started tasks the system allows. That number is usually the MAXUSER value plus the RSVSTRT value. If supervisor recovery reconstructs the ASVT, the maximum number might be the sum of the MAXUSER, RSVSTRT, and RSVNONR values. The default MAXUSER value is still 256.

Removing TRACE Commands from COMMNDxx PARMLIB Members

You might want to remove or update any TRACE operator commands in the COMMNDxx PARMLIB member. The syntax of the TRACE command is changed. MVS/370 TRACE commands do not work in MVS/XA. Also, the MVS/XA system trace remains active after system initialization. No TRACE ON command is required to keep it active, as in MVS/370. Issuing MVS/370 TRACE commands, however, does not prevent MVS/XA system trace from being initialized or activated. "Summary of New, Updated, or Deleted Commands" in Chapter 4 describes the TRACE command changes.

Updating the IEAFIXxx PARMLIB Member

Remove from IEAFIXxx the names of modules that have been moved from LPA to the nucleus. Module names to be removed include:

- IGC0004F (the TTIMER service routine, renamed IGC046 in MVS/XA)
- IGC0004G (the STIMER service routine, renamed IGC047 in MVS/XA)
- IEWFETCH (program fetch, aliases IEWMBOSV and IEWMSEPT)
- IGC0001F (the PURGE service routine)

If those modules are in IEAFIXxx, the operator receives a message indicating that the modules could not be found. The PARMLIB member is not rejected.

Removing References to Device Allocation Tables (IEALPAxx)

The DEVNAMET, IEFDEVPT, and DEVMASKT device allocation tables are deleted in MVS/XA. Remove any references to these tables in PARMLIB members (for example, in the MLPA list).

Keeping RNLs in GRSRNLxx PARMLIB Members

Beginning with Release 2.1.2, you can keep global resource serialization resource name lists (RNLs) in new GRSRNLxx PARMLIB members instead of in the ISGGRNL0 load module in SYS1.LINKLIB. RNLs are easier to update when kept in PARMLIB members. You may, however, continue using the RNLs in SYS1.LINKLIB.

Regardless of where the RNLs are located, if your system is to be part of a global resource serialization complex (GRS=START or GRS=JOIN), you must have at least one GRSRNLxx member. Use the new system parameter, GRSRNL=, to specify which members the system is to use.

If you keep RNLs in SYS1.LINKLIB, the GRSRNLxx member must begin with a statement that tells the system to use the RNLs in SYS1.LINKLIB (and ignore the rest of the statements in the member). That statement is:

```
RNLDEF LINKLIB(YES).
```

To keep RNLs in a GRSRNLxx member, you need to include in the member one statement for each RNL entry. Each statement begins with RNLDEF, specifies the resource name, and indicates the RNL to which it belongs. *MVS/XA SPL: Initialization and Tuning* describes how to write RNLDEF statements.

Beginning with Release 2.1.2, one member, GRSRNL00, is shipped with the release. IEASYS00 contains the parameter GRSRNL=00, so the system uses GRSRNL00 by default.

GRSRNL00 contains entries for the same resources that are in the default RNLs in SYS1.LINKLIB (which are also shipped with the release). In addition, it begins with the statement RNLDEF LINKLIB(YES), which causes the system to use the RNLs in SYS1.LINKLIB.

Because of the defaults, if you are using RNLs in SYS1.LINKLIB, you need not do anything. To use the RNLs in GRSRNL00, you need to:

- Remove the first statement: RNLDEF LINKLIB(YES)
- Add, delete, or modify RNLDEF statements to match your installation's resource serialization requirements.

You can also create and use other GRSRNLxx members.

Systems in the same global resource serialization complex can use different methods of defining RNLs (either statements in GRSRNLxx PARMLIB members or the ISGGRNL0 LINKLIB module). However, as before, the RNLs for all systems in the complex must be identical. The resources identified in the RNLs must be the same and they must appear in the same order.

Specifying Missing Interrupt Handler (MIH) Intervals (IECIOSxx)

Installations can specify by device the time intervals at which MIH scans for missing interrupts. A new MIH statement in the IECIOSxx PARMLIB member allows installations to specify separate time intervals for:

- All DASD except 3330V devices. The default is 15 seconds.
- 3330V devices (MSS virtual DASD). The default is 12 minutes.
- 3851 devices (mass storage controller). The default is 12 minutes.
- Specific devices identified by device number. There is no default. Installations can bypass MIH processing for specific devices by setting a time interval of zero.
- All other devices. The default is 3 minutes.

New, Updated, or Deleted PARMLIB Members

Figure 2-3 summarizes SYS1.PARMLIB members that are new, updated, or deleted in Release 2.1.x. See the *MVS/XA Conversion Notebook, Volume 2* for a summary of the Release 2.2.x SYS1.PARMLIB updates.

Some of the updates are compatible, some are not. For example, if the MVS/370 version of IEASYSxx specifies the ALT parameter, you cannot use it in place of the MVS/XA version of IEASYSxx. (See the entry for IEASYSxx.) In other cases, MVS/XA ignores parameters that it no longer supports and uses defaults for new parameters. If you use the MVS/370 IEAIPSxx member, you need to review the specifications to ensure optimal performance.

MVS/XA SPL: Initialization and Tuning describes the PARMLIB members in more detail.

Member	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
ADYSETxx		x						A new member. It contains records that control dump analysis and elimination (DAE). Each record can specify: - Whether or not DAE is to be active - The functions DAE is to perform - The number of symptom records to be kept in the SYS1.DAE data set See "Dump Analysis and Elimination (DAE)" in Chapter 6 for more information about DAE processing.
CONFIGxx	x							New and changed parameters: - CHAN and CHANNEL are deleted. MVS/XA does not use channel set information. If CHAN and CHANNEL are specified, the operator receives a message indicating that they are ignored. MVS/XA continues processing CONFIGxx. - CHP specifies the configuration of channel paths. It replaces CHAN and CHANNEL. - CPU specifies a processor and can be used in place of CPUAD. You can continue to specify the CPUAD parameter, however. - The syntax of the DEV parameter is changed. It specifies channel path identifiers instead of channel set IDs. The operator can use the CONFIGxx member when reconfiguring the system. The new CONFIG command has an operand, MEMBER, which specifies a CONFIGxx member. In response to a CONFIG MEMBER command, the system logically and physically reconfigures processors, storage, and channel paths as defined by the CPU, STOR, and CHP parameters in the specified CONFIGxx member.
					x			The VF parameter and VFON and VFOFF operands on the CPU parameter can be used in the CONFIGxx member to reconfigure the Vector Facility.
							x	The ESTOR parameter can be used in the CONFIGxx member to reconfigure extended storage elements. As of Release 2.1.7 the order of processing for CONFIG commands specified in the CONFIGxx PARMLIB member changes to become: ONLINE: real storage, extended storage, VFs, CPUs, CHPs OFFLINE: CHPs, VFs, CPUs, extended storage, real storage **Note:** The convention used for numbering resources in the IBM 3090 model 400 processor complex differs from the convention used for numbering resources in the IBM 3084 processor complex. Use the DISPLAY M=SIDE command or refer to Figure 4-2 on page 4-9 to see which processors are on which side of a partitioned processor complex.
GRSRNLxx			x					A new member that contains either global resource serialization resource name lists (RNLs), or a statement indicating the system is to use the RNLs in SYS1.LINKLIB. (Beginning with Release 2.1.2, you can keep RNLs in GRSRNLxx members or in SYS1.LINKLIB.) IBM provides one default member, GRSRNL00. See "Keeping RNLs in GRSRNLxx PARMLIB Members" earlier in this chapter for more information.
GTFPARM	x							Contains new options for requesting I/O event recording. USRP is also a new option, which prompts for specific USR events to be recorded.

Figure 2-3 (Part 1 of 6). New, Updated, or Deleted PARMLIB Members

Member	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
IEAABD00 (SYSABEND)	x							New and changed options on the SDATA keyword: - ALLSDATA includes all of the SDATA options except ALLVNUC and NOSYM. - ALLVNUC requests the PSA, CVT, and DAT-on nucleus. - NOSYM requests that symptom dumps be suppressed. Unless NOSYM is specified, the system produces a symptom dump each time a task abends, even if the user did not specify a dump DD statement. - NUC requests only the DAT-on, non-page protected section of the nucleus and the PSA and CVT. In MVS/370, NUC causes the entire nucleus to be dumped. - SUM requests a summary dump. - TRT requests trace data from the active trace facilities, as in MVS/370. However, only an MVS/XA dump can include both system trace and GTF data. Both trace facilities can be active at the same time in MVS/XA, but not in MVS/370. SUBTASKS, a new option on the PDATA keyword, specifies that the requested PDATA information be dumped for all subtasks of the abending task. When a task receives ABEND code 'x22', the system dumps SUBTASKS data, regardless of whether the SUBTASKS option is specified. The PDATA default options specified in the IBM-supplied member are changed. The MVS/XA default options are all of the PDATA options except SUBTASKS. In MVS/370, ALLPDATA is the default. The IBM-supplied member includes an additional SDATA default option, SUM. It also includes the options specified in the MVS/370 member (LSQA, CB, ENQ, TRT, ERR, DM, and IO).
IEABLDxx	x							Systems with Release 2.1.0 installed process any IEABLDxx member you supply, but provide no default members. Before using existing BLDL lists, ensure their accuracy. Some system modules are moved to different libraries in MVS/XA.
		x						Systems with Release 2.1.1 installed do not process IEABLDxx members. The LNKLST lookaside (LLA) function in Release 2.1.1 or subsequent releases provides a directory of modules in the LNKLST concatenation. The new directory eliminates the need for the BLDL table and, thus, the need for IEABLDxx members. The system ignores the BLDL and BLDLF system parameters. For more information about the LLA function, see "Using a New Directory for LNKLST Data Sets" in Chapter 8.
IEACMD00	x							A new member that contains IBM-supplied commands. Except for one CHNGDUMP command, all of the commands in the Release 2.1.0 member are SLIP commands. The CHNGDUMP command adds trace table and LSQA information to SVC dumps. The SLIP commands suppress dumps that are normally not required for problem determination. See "Suppressing Dumps" in Chapter 6.
		x						Release 2.1.1 adds two new commands: - SET DAE=00, which causes the system to process the ADYSET00 PARMLIB member. ADYSET00 starts DAE processing. For more information, see "Dump Analysis and Elimination (DAE)" in Chapter 6. - START LLA, which starts the LLA procedure in SYS1.PROCLIB. The LLA procedure in turn starts the LNKLST lookaside (LLA) function. See "Using a New Directory for LNKLST Data Sets" for a description of the LLA function.
IEADMP00 (SYSUDUMP)	x							The new and changed parameters for IEADMP00 are the same as those for IEAABD00. See the IEAABD00 entry in this table. The only default option specified in the IBM-supplied member is SUM. In most cases, the summary dump will be sufficient to debug user program checks and ABEND dumps.

Figure 2-3 (Part 2 of 6). New, Updated, or Deleted PARMLIB Members

Member	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
IEADMR00 (SYSMDUMP)	x							New and changed SDATA options are: - ALLNUC, which requests that the entire nucleus be included in the dump. *Note:* You can also request that the entire nucleus be included in SYSMDUMPs via the SNAP parameter list and the CHNGDUMP command. The required SNAP parameter list option is ALLVNUC, not ALLNUC. The CHNGDUMP option is ALLNUC. - ALLSDATA includes all of the SDATA options except ALLNUC and NOSYM. - NOSYM, NUC, and TRT request the same data as when specified in IEAABD00. See the IEAABD00 entry in this table. - SUM requests summary dump information like that included in SVC dumps. The default options specified in the IBM-supplied member include SUM, in addition to the SDATA options specified in the MVS/370 default member (NUC, SQA, LSQA, SWA, TRT, and RGN).
IEAFIXxx	x							Unless the NOPROT option is specified on the FIX parameter in the IEASYSxx member, the system page-protects the modules listed in IEAFIXxx. See "Page Protection" in Chapter 3.
IEAIPSxx	x							A new parameter, IOSRVC, specifies whether SRM is to base I/O service on EXCP counts or device connect time. The default is EXCP counts.
			x					A new parameter, PPGRTR, requests that SRM use residency time instead of execution time when it calculates the page-in rate for address spaces in the specified performance group. PPGRTR specifies the high or low limit the rate must exceed before SRM adjusts the address space's working set size. See "Using Residency Time to Calculate the Page-in Rate of an Address Space" in Chapter 8 for more detail.
							x	The Rx parameter in the DP keyword has no meaning. When it is encountered, the system accepts it as Fx, the new first fixed priority. As "Automatic Priority Group (APG) Specifications" on page 8-11 explains, the fixed priority specifications are now: Fx, Fx0, Fx1, Fx2, Fx3, and Fx4.
IEALPAxx	x							Unless the new NOPROT option is specified on the MLPA parameter in the IEASYSxx member, the system page-protects the modules listed in IEALPAxx. See "Page Protection" in Chapter 3.
IEALOD00	x							If IEALOD00 is specified, MVS/XA ignores it. Unlike MVS/370, MVS/XA does not build contents directory entries (CDEs) for PLPA modules. It uses the LPA directory entries (LPDEs) instead.

Figure 2-3 (Part 3 of 6). New, Updated, or Deleted PARMLIB Members

Member	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
IEAOPTxx	x							New parameters: - CPENABLE specifies upper and lower thresholds for the percentage of I/O interrupts that occur while the channel subsystem is processing another I/O interrupt. SRM uses the thresholds to control I/O interrupt processing. - ICCLPB(TAPE), ICCLPB(NDPSDASD), and ICCLPB(DPSDASD) specify logical path utilization thresholds for tape, DASD without the dynamic path selection feature, and DASD with the dynamic path selection feature, respectively. SRM uses the thresholds for I/O load balancing and for non-specific device allocation.
				x				Release 2.1.3 adds eight new SRM parameter keywords that represent criteria table values SRM uses to determine whether or not a page is placed on extended storage. Only in rare instances is there a need to change these values. See "Expanded (Extended) Storage Criteria" in Chapter 5.
IEAPAKxx	x	x						MVS/SP Version 2 does not provide PAK lists. However, it processes IEAPAKxx members that you supply. Release 2.1.0 and earlier releases recognize only IEAPAK00. Release 2.1.1 and later releases allow multiple IEAPAKxx members. By using more than one member, you can vary the LPALST concatenation from system to system or from IPL to IPL without changing IEAPAK00. The PAK=xx system parameter specifies which members the system is to use. IEAPAK00 is the default, which is consistent with earlier releases. Do not use existing PAK lists without first reevaluating their usefulness. Unless all modules in a group have the same RMODE (that is, they all reside in virtual storage either above or below 16 megabytes), IPL/NIP ignores the PAK list.
IEASYSxx	x							New, changed, and deleted parameters: - CMB specifies the I/O device classes for which measurement data is to be collected. The CMB specifications are in addition to DASD and tape device classes, for which measurement data is always collected. - The ALT parameter is no longer supported. Have operators use the SYSCTL console frame to specify an alternate nucleus. If ALT is specified, MVS/XA truncates processing and asks the operator for another member. Because some system parameters might already have been processed, have operators re-IPL and request an IEASYSxx member that does not contain the ALT parameter. - The MLPA and FIX parameters have an additional option, NOPROT. NOPROT indicates that the LPA modules listed in the IEALPAxx or IEAFIXxx PARMLIB member are not to be page-protected. Unless the NOPROT option is specified, the system page-protects those modules. See "Page Protection" in Chapter 3. - A new option on the DUMP parameter, 'DASD,xx-yy', specifies which currently cataloged SYS1.DUMPnn data sets the system is to use. If none are specified, the system uses any that are cataloged. See "Specifying Dump Data Sets (IEASYSxx)" for more information. - CSA has an additional option that specifies the size of CSA above 16 megabytes. - SQA has an additional option that specifies the size of SQA above 16 megabytes. For more information about the CSA and SQA parameters, see " Specifying the Size of Extended CSA and Extended SQA."

Figure 2-3 (Part 4 of 6). New, Updated, or Deleted PARMLIB Members

Member	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
IEASYSxx (continued)		x						New parameters: - LNKAUTH specifies whether all data sets in the LNKLST concatenation are to be treated as APF authorized or only those named in the APF table. The default is to treat all of the data sets as APF authorized, as do previous releases of MVS. See "Using a New Directory for LNKLST Data Sets" for more information. - LPA identifies the LPALSTxx member to be processed. See the LPALSTxx entry in this table for more information. - MSTRJCL specifies which MSTJCLxx member in SYS1.LINKLIB the system is to use. If you omit the MSTRJCL parameter, the system uses the JCL in the IBM-supplied default, MSTJCL00. MSTJCLxx members are new in Release 2.1.1. For more information, see "MSTJCLxx Members in the SYS1.LINKLIB Data Set," under "Defining System Data Sets" on page 2-8. - PAK identifies the IEAPAKxx member to be processed. See the IEAPAKxx entry in this table for more information. The BLDL and BLDF parameters are obsolete. If specified, the operator receives warning message IEA240I. The LNKLST lookaside (LLA) function provides a directory of modules in the LNKLST concatenation. The new directory eliminates the need for an IEABLDxx member, and consequently, the BLDL and BLDF parameters. For more information, see "Using a New Directory for LNKLST Data Sets" in Chapter 8.
			x					New and changed parameters: - GRSRNL specifies which GRSRNLxx member the system is to process. If your system is to be part of a global resource serialization complex, you must specify a value for GRSRNL. It has no default. Beginning with Release 2.1.2, the IEASYS00 member includes the statement GRSRNL=00. However, unless your installation performs a system generation to install Release 2.1.2, your copy of IEASYS00 is not updated. You need to add the GRSRNL statement yourselves. (The system generation process creates IEASYS00. No other methods of installation modify it.) See "Keeping RNLs in GRSRNLxx PARMLIB Members" for more information. - RSVSTRT specifies the number of ASVT entries ASM is to reserve for address spaces created in response to a START command. ASM uses these reserve entries only if no unreserved ASVT entries are available. The default value is five. - RSVNONR specifies the number of ASVT entries ASM is to reserve as replacements for entries it cannot reuse. ASM uses the replacements only if it runs out of unreserved ASVT entries. The default is five. - MAXUSER still limits the number of jobs and started tasks that can execute concurrently under normal conditions. However, it no longer specifies the maximum number of jobs or started tasks the system allows. Beginning with Release 2.1.2, that number is normally the MAXUSER value plus the RSVSTRT value. The default MAXUSER value is still 256. The last three parameters are related to changes described in "Controlling the Number of Available ASVT Entries (IEASYSxx)."

Figure 2-3 (Part 5 of 6). New, Updated, or Deleted PARMLIB Members

Member	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
IECIOSxx	x							The LCH parameter is no longer supported. A new MIH control statement specifies intervals at which MIH is to scan for MIH conditions. MIH allows installations to specify different time intervals for different devices and/or types. See "Specifying Missing Interrupt Handler (MIH) Intervals (IECIOSxx)."
		x						A new statement, HOTIO, allows you to change: - The threshold IOS uses to detect hot I/O conditions - The recovery actions IOS takes when it detects a hot I/O condition See "Processing Hot I/O Interrupts" in Chapter 4 for more information.
LNKLSTxx		x						The data sets in the LNKLST concatenation no longer have to be APF authorized. Also, you can concatenate up to 123 data sets. Earlier releases allow no more than 16. See "Using a New Directory for LNKLST Data Sets" in Chapter 8 for more information.
LPALSTxx		x						A new member that lists data sets to be concatenated to the SYS1.LPALIB data set. See "Concatenating Data Sets to the SYS1.LPALIB Data Set" under "Defining System Data Sets" for more information.
MPFLSTxx	x							Release 2.1.0 adds two new keywords: - .MSGCOLR specifies how messages are to be displayed on MCS color consoles that can use seven colors and other forms of highlighting. An MPFLSTxx member can also include the statement .MSGCOLR NOCHANGE, which maintains the color and highlighting attributes in effect when the member became active. - .MSGIDS NOCHANGE requests that the system not change the message IDs already in effect.
			x					Release 2.1.2 adds new options on.MSGCOLR statements and on message suppression records. The new option on .MSGCOLR specifies whether the message area is to be displayed in normal or high intensity. On message suppression records, you can specify: - SUP(YES|NO) to control whether or not the messages are suppressed. The default is SUP(YES) to maintain compatibility with previous releases. - RETAIN(YES|NO) to control whether or not the messages are to be retained via the action message retention facility. RETAIN(YES) is the default. - USEREXIT(NAME) specifies the name of a WTO/WTOR user exit the system is to call each time it issues the messages. WTO/WTOR user exits are new in Release 2.1.2. For more information, see "New WTO/WTOR User Exits" in Chapter 5.
SMFPRMxx		x						Release 2.1.1 and subsequent releases ignore the BUFNUM parameter. The SMF buffers are moved to the new SMF address space. Consequently, the number of buffers SMF can obtain is limited only by the amount of virtual storage in the SMF address space. SMF initially obtains 10 buffers and requests more as needed. If the number of buffers in use exceeds 125, SMF informs the operator via message IEE978E.

Figure 2-3 (Part 6 of 6). New, Updated, or Deleted PARMLIB Members

SYS1.PROCLIB Changes

The SYS1.PROCLIB data set in Release 2.1.0 contains a new DUMPSRV procedure and a new statement in the PRDMP procedure. The RMF procedure in RMF Version 3 is also changed. Release 2.1.1 adds two new procedures, LNKLST Lookaside (LLA) and IEESYSAS. Release 2.1.2 changes the PRDMP procedure. Release 2.1.3AE adds the AVM procedure. Either use the SYS1.PROCLIB shipped with the product, or copy the new and changed procedures into your version of SYS1.PROCLIB.

Following are descriptions of the new and changed procedures. If your MVS/XA system is part of a loosely-coupled JES3 configuration that includes MVS/370 systems, also read "Using SYS1.PROCLIB in a Loosely-Coupled JES3 Configuration" in Chapter 9.

AVM Procedure

The AVM procedure starts the availability manager address space as an MVS/XA started task:

```
//AVM  EXEC PGM=AVFMNBLD,MODE=OPERATOR
```

DUMPSRV Procedure

The DUMPSRV procedure starts the dump service (DUMPSRV) address space:

```
//DUMPSRV EXEC PGM=IEAVTDSV
```

IEESYSAS Procedure

IEESYSAS is a new procedure in Release 2.1.1 that starts full-function system address spaces, which include the SMF address space.

```
//IEESYSAS PROC PROG=IEFBR14
//     EXEC PGM=&PROG
```

LNKLST Lookaside (LLA) Procedure

The LLA procedure starts the LNKLST lookaside (LLA) function:

```
//LLA  EXEC PGM=CSVLLCRE
```

Beginning with Release 2.1.1, IEACMD00 PARMLIB member contains a START LLA command, which starts the LLA procedure. See "Using a New Directory for LNKLST Data Sets" in Chapter 8 for more information about the LLA function.

PRDMP Procedure

Releases 2.1.0 and 2.1.2 change the PRDMP procedure:

In Release 2.1.0 there is a new INDEX DD statement. This statement requests that PRDMP write the dump index to a sequential data set other than the PRINTER data set. If the INDEX DD statement precedes the PRINTER DD statement, the index is printed before the dump. If the INDEX DD statement is missing, PRDMP prints the index on the PRINTER data set after the dump.

In Release 2.1.2 PRDMP runs as a command processor under TSO. As a result, the EXEC statement has been changed and three DD statements, SYSPRINT, SYSTSIN, and SYSTSPRT, are now required. You can, however, specify dummy DD statements for any of the three. The INDEX DD statement is new in Release 2.1.0.

As of Release 2.1.2 the PRDMP procedure is:

```
//PRDMP     PROC DUMP=DUMP00
//DMP     EXEC PGM=IKJEFT01,PARM=AMDPRDMP
//SYSTSIN   DD DUMMY,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSTSPRT DD DUMMY
//SYSPRINT DD SYSOUT=A
//TAPE     DD DSN=SYS1.&DUMP,DISP=SHR
//INDEX     DD SYSOUT=A
//PRINTER   DD SYSOUT=A
//SYSUT1    DD UNIT=SYSDA,SPACE=(4104,(1027,191))
```

The EXEC statement must invoke IKJEFT01, the TSO terminal monitor program. Specifying PARM = AMDPRDMP on the EXEC statement causes IKJEFT01 to invoke PRDMP. You can also specify on PARM additional parameters for AMDPRDMP, as in previous PRDMP procedures. The valid parameters are described in *SPL: Service Aids.*

SYSPRINT, which previously was optional, directs system messages (except those IKJEFT01 issues) to the specified data set. If you use a dummy statement, the system does not log those messages.

SYSTSIN and SYSTSPRT are both DD statements that IKJEFT01 uses. SYSTSIN specifies a data set that contains commands or subcommands IKJEFT01 is to execute. SYSTSPRT identifies the data set in which the system is to log messages that IKJEFT01 issues. If you specify a dummy SYSTSPRT DD statement, the system does not log the messages.

RMF Procedure

RMF Version 3 contains a new procedure for starting RMF. The new procedure adds one statement:

```
RMF Version 3 procedure:
// ...
// ...
// ...
//IEFPARM  DD  DDNAME=IEFRDER        (new statement)
//IEFRDER  DD  DSN=SYS1.PARMLIB,DISP=SHR
```

The IEFPARM statement must come before IEFRDER.

The new statement is required because RMF Version 3 opens IEFPARM, not IEFRDER, as does RMF Version 2. The IEFRDER statement is necessary to allow operators to override or specify additional options on IEFRDER via the START command. (See the *RMF Reference and User's Guide* for details.)

The RMF Version 3 procedure cannot start RMF Version 2 because the Version 2 procedure opens IEFRDER. When processing the Version 3 procedure, the system associates the data set information on the IEFRDER statement with the IEFPARM ddname, then deletes the IEFRDER name. Therefore, if MVS/370

executes the Version 3 procedure, RMF cannot find the IEFRDER statement when it attempts to open SYS1.PARMLIB.

You can, however, modify the IEFPARM statement in the Version 3 procedure to obtain a procedure that can start either RMF Version 2 or 3. Replace the IBM-supplied IEFPARM statement with the following one:

```
// ...
// ...
// ...
//IEFRDER DD DSN=SYS1.PARMLIB,DISP=SHR
//IEFPARM DD DSN=*.IEFRDER,DCB=*.IEFRDER,
      VOLUME=REF=*.IEFRDER,DISP=SHR
```

Notice that, unlike the original IEFPARM statement, the modified statement must come *after* the IEFRDER statement. You can specify additional data set information on IEFRDER. However, you must also specify the same keywords and values on IEFPARM or the system ignores the information.

Duration of the RMF Initialization Process

The first RMF initialization following an IPL takes up to a minute longer in MVS/XA than in MVS/370. The increase represents the time required for the initialization routines to obtain configuration information from the IOCDS. The MVS/370 initialization routines do not use comparable information.

Using Default RNLs

Beginning with Release 2.1.2, you can IPL a system without modifying the IBM-supplied resource name lists (RNLs). Previously, if you specified GRS=JOIN or GRS=START, you had to include entries in the SYSTEMS exclusion RNL for global RESERVE requests that the system issues during IPL processing (for example, the RESERVE request for the system master catalog). If earlier releases encounter a global RESERVE request before global resource serialization is initialized, and the request is not in the SYSTEMS exclusion RNL, the system stops. (Global ENQs are not mentioned here because the system issues none before resource serialization is initialized.)

Release 2.1.2 IPL processing treats global RESERVE requests as local requests until global resource serialization is initialized. For each global RESERVE processed, the system issues message ISG066I, which states that the resource is temporarily excluded from global processing.

In the following situations, however, you still need to add entries to the default SYSTEMS exclusion RNL:

- Other systems in the global resource serialization complex have additional entries in the RNL (for example, systems at earlier levels of MVS, which require entries in the SYSTEMS exclusion RNL). The RNLs of all systems in the complex must be identical.
- Some global RESERVE requests are to be treated as local requests after global resource serialization is initialized.

Generating Stand-Alone Dump

You can generate the stand-alone dump program (SADMP) and initialize the volumes on which it resides in one batch job instead of two, as required in MVS/370. The old two-step procedure still works. *SPL: Service Aids* describes how to perform the same functions in one step. You must use Assembler H Version 2 to generate MVS/XA SADMP.

MVS/SP Version 2 introduces several other improvements to stand-alone dump that might affect how you code the AMDSADMP macro instruction. For example, to ensure that you successfully load stand-alone dump, you must specify usable addresses for loading the real storage dump module. The new LOADPT keyword makes this possible.

The section below describes the changes to the AMDSADMP macro. The *SPL: Service Aids* describes how to code them.

Stand-Alone Dump Macro (AMDSADMP) Changes

Beginning with MVS/SP Version 2, the stand-alone dump macro (AMDSADMP) has several new and changed keywords that improve its usefulness. The following figure summarizes the changes:

Keyword	Status	Description of Change
CONSOLE	Changed	Accepts 2 to 21 device addresses and device types in MVS/XA. The default device type is also changed. The MVS/XA default is 3278, the MVS/370 default is 3215. The default device address (01F) remains the same. In MVS/XA, CONSOLE must include the addresses of all consoles that SADMP can use. Also, after performing an SADMP IPL, the MVS/XA operator must press the ENTER or ATTN key on the console which SADMP is to use.
DUMP	New	Allows the user to select storage areas to be dumped in an unformatted dump. The areas specified are **in addition to** the areas that a stand-alone dump normally includes. DUMP is valid only for high speed stand-alone dumps.
LOADPT	New	Specifies an absolute address where the stand-alone dump real storage dump module (AMDSARDM) is to be loaded. LOADPT allows users to avoid bad or offline storage.
MSG	New	Specify MSG=ACTION to request that SADMP display only messages that require action. If MSG=ACTION is not coded, SADMP displays both information and action messages. Suppressing information messages speeds up dump processing.
PROMPT	New	Requests that SADMP prompt the operator at execution time for additional storage areas to be dumped. PROMPT provides the same function as the DUMP keyword, but allows the operator to make storage requests at the time a dump is taken instead of when SADMP is generated. You can specify PROMPT on the same macro as DUMP. Like DUMP, PROMPT is valid only for high speed stand-alone dumps.

Figure 2-4. Stand-Alone Dump Macro Instruction Changes

Chapter 3. Programming Considerations

This chapter describes differences that might affect user-written assembler programs, including user modifications to the system. For Releases 2.1.0, 2.1.1, and 2.1.2, the updates are grouped according to the type of programs each might affect. For subsequent releases, there are few conversion issues that affect programming. They are summarized in the following section.

Programming Considerations Subsequent to Release 2.1.2

- For Release 2.1.3, there are two programming considerations:

  1. The ADDFRR instruction is no longer supported.
  2. The SRM control block, OUCB (IRAOUCB), is updated incompatibly.

- Release 2.1.3VFE and Release 2.1.3AE are mutually exclusive.

  See Chapter 7, "Accounting" for an explanation of the restructure of the SMF type 6 record that is included in Release 2.1.3VFE.

- For Release 2.1.7 there are four programming considerations:

  1. The Release 2.1.7 base control program (BCP) incorporates both Release 2.1.3VFE and Release 2.1.3AE.
  2. The ASCBs on the dispatching queue (which is called the swapped-in queue as of Release 2.1.7) are no longer in dispatching-priority order.
  3. The sequence number field (ASCBSEQN) is deleted from the ASCB. As a result, RMF Version 3 Release 4.1 places a zero in the dispatching queue field of the Type 79 subtype 1 record.
  4. The Installation Channel Path Table (ICHPT) resides above the 16 megabyte virtual address.

Changes that Might Affect Unauthorized Programs

With few exceptions, programs that use only unauthorized services and published external interfaces will work unchanged in MVS/XA. The exceptions include programs that use the following macro instructions:

- IOHALT
- IOSGEN UCBLOOK
- RESETPL
- SPIE (in two circumstances only)
- STATUS with the STOP,SYNCH option specified

You can modify the affected programs before installing MVS/SP Version 2. See the topics describing each macro.

Some unauthorized programs might also require modification because of changes described in "System Diagnostic Work Area (SDWA) Changes" on page 3-11 and "Differences in GETMAIN Processing" on page 3-11.

Topics describing changes that apply to all programs are:

Changes that Might Affect Authorized Programs

Authorized programs are those that execute either in:

- Supervisor state (bit 15 in the PSW is zero)
- A system key (bits 8-11 in the PSW are in the range 0-7)
- As part of an APF-authorized job step task (bit JSCBAUTH in the JSCB is 1)

Although many authorized programs will work unchanged in MVS/XA, you might have to modify some. Those most likely to require modification are programs that:

- Use system interfaces that are not documented externally.
- Communicate directly with system modules (for example, via branch entry)
- Access system control blocks that are changed or that have been moved to virtual storage above 16 megabytes
- Modify the system
- Have dependencies on the names or virtual storage locations of system modules

In addition to topics already mentioned, the following describe changes that might affect authorized programs:

31-bit Addressing Considerations

During the migration period, most users do not need to be concerned with addressing mode. The information in this section is for installations with programs that have any of the following characteristics:

- Access system control blocks that have been moved to virtual storage above 16 megabytes.
- Use unpublished internal interfaces to communicate with system programs that must be entered in 31-bit addressing mode.
- Use RMF Version 3 exits

The following topics describe changes that support 31-bit addressing:

New Function

The information in this section affects only new programs. The topics describe new functions available to both authorized and unauthorized users:

"Chapter 9: Coexistence Considerations" contains additional considerations for maintaining programs that must run in both MVS/XA and MVS/370.

The following topic, "Macro Instructions Mentioned in This Publication," lists the executable macros that might require attention and indicates why.

Macro Instructions Mentioned in This Publication

Figure 3-1 lists the executable macros that might require attention when converting to MVS/XA. The macros are included in the list for one or more of the following reasons:

1. The MVS/XA expansion of the macro will not work in MVS/370 (that is, the macro is downward incompatible). The downward incompatibility affects only programs that must run on both MVS/370 and MVS/XA systems. If such a program uses one of these macros, ensure that the program either:

   - Includes the MVS/370 expansion of the macro
   - Has two paths, one that MVS/370 executes, the other that MVS/XA executes

   For more information, see "Handling Downward Incompatible Macros" in Chapter 9.

2. The MVS/370 macro expansion does not work in MVS/XA. You must reassemble, using the MVS/XA MACLIB, all programs that use these macros. For most of the macros, you can install compatibility PTFs or compatible program products on an MVS/370 system and reassemble the affected programs before installing MVS/XA. See the references in the notes column.

3. The macro expansion passes different parameters to the associated service routine. The parameter changes affect only programs that generate, test, or alter the parameters. You must change those programs. "Appendix A. Parameter Changes in Incompatible Macros" describes the differences between the MVS/370 and MVS/XA parameter lists.

4. The MVS/XA expansion is required if used in programs that execute in 31-bit addressing mode. Thus, you must assemble such programs using Assembler H Version 2 and the MVS/XA MACLIB.

5. The macro provides new function. The list includes macros that are new or that have new parameters or new options on existing parameters. The added function does not affect existing programs. Figure 3-6 summarizes the new function that each macro provides.

   You must use the MVS/XA MACLIB to assemble programs that use new functions. With two exceptions, the programs will then not work on MVS/370 systems. New GETMAIN options are one exception. The new AMODE=24 option on the SYNCH macro is the other. See "Parameters on the GETMAIN Macro Instruction" in this chapter and "Downward Incompatible SYNCH Macros" in Chapter 9 for more detail.

6. The macro requires attention for another reason. The notes column refers to the topic that describes the reason.

System Macros and Facilities documents authorized macros. *Supervisor Services and Macro Instructions* documents unauthorized macros. Some of the macros listed as unauthorized have parameters that only authorized users can specify. Those macros are documented in both publications.

Unauthorized Macros	Incompatible MVS/XA Expansion 1.	Incompatible MVS/370 Expansion 2.	Parameter Changes 3.	31-Bit Mode Requirement 4.	New Function 5.	Other 6.	Notes
ABEND					x		
ATTACH	x		x	x			The MVS/XA expansion is required only if parameter addresses are greater than 16 megabytes.
BLSABDPL					x		
BLSQMDEF					x		
BLSQMFLD					x		
BLSRESSY					x		
BTAM RESETPL		x	x				See "RESETPL (BTAM) Macro Instruction."
CALL				x			The MVS/XA expansion is required only if parameter addresses are greater than 16 megabytes.
CHKPT	x			x		x	See "CHKPT Macro Instruction."
CPOOL					x		
CPUTIMER					x		
ENQ					x	x	See "Limiting Concurrent Global Resource Serialization Requests."
ESPIE					x		
ESTAE	x		x	x			
EVENTS	x		x	x			
EXCP					x		See "Using the EXCP Macro."
GETMAIN					x		See "Parameters on the GETMAIN Macro Instruction."
GQSCAN						x	See "Limiting Concurrent Global Resource Serialization Requests."
GTRACE					x		
LINK				x			The MVS/XA expansion is required only if parameter addresses are greater than 16 megabytes.
IOHALT		x	x				See "IOHALT Macro Instruction (SVC 33)."
IOSGEN UCBLOOK						x	Deleted. See "IOSGEN UCBLOOK Macro Instruction."
LOAD					x		
PGSER					x		
RACROUTE					x		Becomes authorized if used to invoke authorized RACF services.
RESERVE					x	x	See "Limiting Concurrent Global Resource Serialization Requests."

Figure 3-1 (Part 1 of 2). Unauthorized Macro Instructions Mentioned in This Publication

Unauthorized Macros	Incompatible MVS/XA Expansion 1.	Incompatible MVS/370 Expansion 2.	Parameter Changes 3.	31-Bit Mode Requirement 4.	New Function 5.	Other 6.	Notes
RETURN						x	See the entry for RETURN in "Summary of New and Updated Macros"
SETRP					x		
SMFEXIT	x		x	x			
SMFIOCNT					x		
SNAP					x		
SPIE						x	Works differently in some situations. See "Differences in Set Program Interruption Element (SPIE) Processing."
SPLEVEL					x		See "Handling Downward Incompatible Macros" in Chapter 9.
STATUS STOP,SYNCH		x					See "STATUS STOP,SYNCH Macro Instruction."
STAX	x		x	x			The MVS/XA expansion is required only if parameter addresses are greater than 16 megabytes.
STIMER	x		x	x			
STIMERM					x		
SYNCH	x				x	x	See "Downward Incompatible SYNCH Macros" in Chapter 9.
WTL					x		
WTO					x		See the entry for WTO in "Summary of New and Updated Macros."
WTOR	x		x	x	x		
XCTL				x			The MVS/XA expansion is required only if parameter addresses are greater than 16 megabytes.

Figure 3-1 (Part 2 of 2). Unauthorized Macro Instructions Mentioned in This Publication

Authorized Macros	Incompatible MVS/XA Expansion 1.	Incompatible MVS/370 Expansion 2.	Parameter Changes 3.	31-Bit Mode Requirement 4.	New Function 5.	Other 6.	Notes
CALLDISP				x			
CALLRTM					x		
CHANGKEY					x		
CIRB					x		
DATOFF					x		
EXCPVR					x		See "Using the EXCPVR Macro Instruction."
FESTAE	x			x			
IOSINFO				x	x		
IOSLOOK					x		Locates the UCB associated with a device address. See "IOSGEN UCBLOOK Macro Instruction."
INTSECT	x			x			
MODESET				x			The MVS/XA expansion is required only if EXTKEY = RBT234 is specified.
MGCR					x		
NUCLKUP					x		
PTRACE					x		
SCHEDULE	x						Downward incompatible only if SCOPE = GLOBAL is specified.
SDUMP	x		x	x	x		Downward incompatible only if new parameters are specified. See "SDUMP Macro Instruction." See also the entry for SDUMP in "Summary of Macros that Provide New Function."
SETFRR	x						Downward incompatible only if INLINE = YES is specified.
SETLOCK	x		x		x		Downward incompatible only if RELEASE TYPE = (reg)|ALL is specified. See "Determining Which Locks a Processor Holds" for an example of new SETLOCK function.
SVCUPDTE					x		
VSMLIST					x		
VSMLOC					x		
VSMREGN					x		

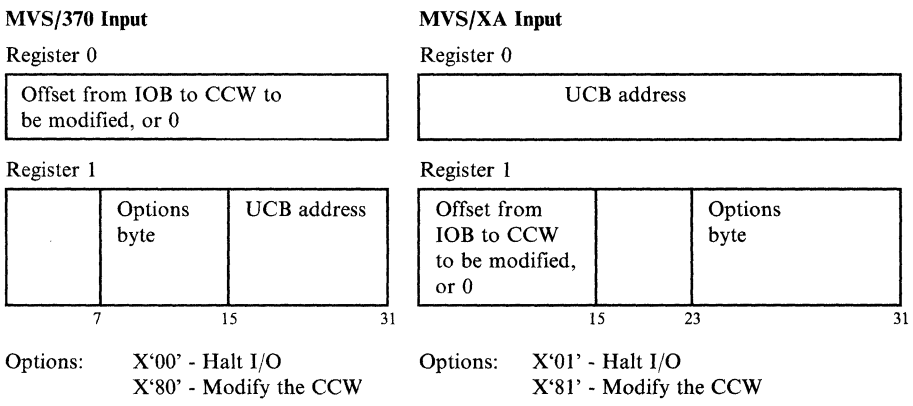
Figure 3-2. Authorized Macro Instructions Mentioned in This Publication

CHKPT Macro Instruction

User programs that successfully take checkpoints in MVS/370 can take checkpoints in MVS/XA. However, a program that has taken a checkpoint must restart on the same operating system (either MVS/370 or MVS/XA).

IOHALT Macro Instruction (SVC 33)

You must recompile modules that use the IOHALT macro and modify programs that issue SVC 33 directly (without using IOHALT). The SVC 33 service routine requires different input in registers 0 and 1:



Options: X'00' - Halt I/O
X'80' - Modify the CCW

Options: X'01' - Halt I/O
X'81' - Modify the CCW

PTF UZ29156 provides the new version of IOHALT and the new SVC 33 interface that are compatible with MVS/XA. You can install the PTF on an MVS/370 system and reassemble and modify the affected programs before installing MVS/SP Version 2. (You do not have to reassemble programs that will not be run on an MVS/XA system.) The reassembled programs will work on both MVS/370 and MVS/XA systems.

If you use GTF to trace modules that use IOHALT or SVC 33 and you install the IOHALT compatibility PTF, you might need to install an additional PTF on MVS/370. Unless you have installed MVS/SP Version 1 Release 1 or a later release, install either:

PTF UZ32985 for systems with SE2 installed
PTF UZ32984 for systems with SE1 installed
PTF UZ32983 for MVS 3.8 systems with neither SE1 nor SE2 installed

The PTFs allow GTF to trace programs that use either the old or the new SVC 33 interface. MVS/SP Version 1 Release 1 and later releases incorporate the PTF changes.

IOSGEN UCBLOOK Macro Instruction

You must change programs that use the IOSGEN UCBLOOK macro or that directly access the UCB look-up table. Neither the IOSGEN UCBLOOK function nor the UCB look-up table is supported in MVS/XA.

To obtain UCB addresses in MVS/XA, use either:

- The UCB scan routine (IOSVSUCB)
- The IOSLOOK macro

IOSVSUCB allows you to scan each UCB in the system or in a specified device class. Each time it is invoked, IOSVSUCB returns the address of one UCB's common segment. To scan several UCBs, invoke IOSVSUCB repeatedly. Both authorized and unauthorized programs can use IOSVSUCB.

IOSLOOK returns the address of the common segment of the UCB associated with a given device number. Unlike IOSVSUCB, IOSLOOK requires that users be in supervisor state.

Both IOSVSUCB and IOSLOOK are documented in *System Macros and Facilities.*

Both services are also available in MVS/370. MVS/SP Version 1 Release 3 and later releases include IOSVSUCB. PTF UZ28392 includes IOSLOOK. Therefore, you can change the affected programs before installing MVS/XA. The changed programs will run on both MVS/370 and MVS/XA.

RESETPL (BTAM) Macro Instruction

Programs, including CICS and IMS BTAM modules, that use the OS/VS BTAM expansion of RESETPL will not work in MVS/XA. You must reassemble them using the RESETPL macro included in BTAM/SP (5665-279). (BTAM/SP is required to run BTAM application programs and subsystems in MVS/XA.)

You can install BTAM/SP on MVS/370 and reassemble the affected programs before installing MVS/XA. The reassembled programs will work on both MVS/370 and MVS/XA systems.

Note: Because the RESETPL expansion issues an IOHALT macro, you must also install the IOHALT compatibility PTF on MVS/370 before reassembling the programs. See "IOHALT Macro Instruction (SVC 33)" for more detail .

Differences in Set Program Interruption Element (SPIE) Processing

Most programs using SPIE macros will continue to work correctly in MVS/XA. However, you need to modify programs that create a SPIE to protect a program running under a different RB.

MVS/XA terminates the SPIE when the program that created it completes, whether normally or abnormally. In MVS/370, the SPIE usually remains in effect until all programs in the step complete (task termination time). The exception occurs when the program that creates the SPIE abends. If that happens, MVS/370 terminates the SPIE also.

Following are two examples of programs that do not work the same in MVS/XA as in MVS/370:

- Program A links to Program B, which issues a SPIE macro and returns. In MVS/XA, the SPIE is deleted. In MVS/370, it remains in effect.

- Program A issues a SPIE macro, followed by an XCTL macro to invoke Program B. In MVS/XA, the SPIE is deleted. In MVS/370, the SPIE is in effect for Program B.

You can change affected programs before installing MVS/XA.

STATUS STOP,SYNCH Macro Instruction

The SYNCH operand on the STATUS STOP macro is no longer supported. Change programs that issue STATUS STOP,SYNCH to issue STATUS STOP without the SYNCH operand. You can make the changes before installing MVS/XA.

System Diagnostic Work Area (SDWA) Changes

The SDWA is increased in size. The additional storage is included in previously existing or new SDWA extensions. The sizes of the FRR work area and the ESTAE save area remain the same.

Programs that use indirect pointers into the SDWA work unchanged in MVS/XA. However, you must modify programs that:

- Depend on the 72-byte save area passed to ESTAE exits being located at a given offset into the SDWA. MVS/XA uses register 13 to pass to ESTAEs the address of the user save area, as does MVS/370.
- Depend on the 200-byte FRR work area that is passed to FRR routines being located at a given offset into the SDWA. MVS/XA uses Register 0 to pass to FRRs the address of that work area, as does MVS/370.
- Use explicit length values to free the SDWA. Modify the programs to use the value in the SDWALNTH field.
- Depend on the order of the SDWA and its extensions. Modify the programs to use indirect pointers.
- Place data in the SDWA variable recording area (VRA) without updating the SDWAURAL field. Programs must maintain an accurate count in the SDWAURAL field to prevent data from being overlaid.
- Assume that the unused section of the SDWA contains zeros. Programs need to ignore data in the unused area.

Differences in GETMAIN Processing

Two differences in GETMAIN processing might cause programs to fail in isolated instances:

- Although the MVS/XA GETMAIN service routine does not introduce any new parameter restrictions, it does enforce some restrictions that were documented but not enforced in MVS/370. With one exception, the GETMAIN routine no longer allows the parameter list, address list, or length list specified on the LC, LU, VU, EC, or EU forms of GETMAIN to overlap. If the request is for a single element, MVS/XA allows the pointer to the address list to point to itself. All other overlaps cause the program to fail with ABEND code x'504'.

Because programs seldom use the forms of GETMAIN mentioned or overlap parameters, you probably will not want to spend time looking for programs that have to be changed. Instead, keep in mind the parameter restrictions. If a program fails with ABEND code x'504', you can change it then.

- GETMAIN routines obtain storage differently in MVS/XA. If your programs expect that additional virtual storage will be contiguous with currently owned virtual storage, you need to modify them. Additional storage might not be contiguous.

TSO TEST Command

To use TSO TEST on an MVS/XA system, you must install the MVS/XA feature of TSO/E (5665-285). TSO TEST is not part of the MVS/XA base control program. If you issue TSO TEST and have not installed TSO/E for MVS/XA, you receive message IKJ56500I stating that the TEST command is not found. In addition, unless TSO/E for MVS/XA is installed, user programs that issue either SVC 61 or SVC 97 receive a return code of 4.

Following are TSO TEST compatibility considerations:

- You can test programs created in MVS/XA on MVS/370 as long as the programs do not use any MVS/XA instructions, new macros, new parameters or options on existing macros, downward incompatible macros, or addresses above 16 megabytes.

- When executing TSO TEST on an MVS/XA system, AT subcommands and LIST subcommands that specify the instruction data type support only MVS/XA instructions.

- User-written TEST subcommands that do any of the following will not work with the MVS/XA version of TSO TEST:

  - Use IKJEGSTA as an ESTAE exit. The parameter list that IKJEGSTA requires is incompatible.

  - Use the TCOMTAB mapping macro to access fields in the TCOMTAB control block. Many labels on the TCOMTAB macro are deleted because TSO TEST no longer uses the corresponding fields.

  - Use labels and equates in TSO TEST mapping macros to determine the length of the corresponding TEST control blocks. The names of the equates used to define the lengths of control blocks are changed.

- Installations that altered the TSO TEST subcommand table (IKJEGSCD) must rebuild their changes in the new table. To update the table:

  - Copy the IKJEGSCD CSECT of assembly module IKJEGMNL in the TEST load module into a separate data set.

  - Make the required changes.

  - Assemble and again link edit IKJEGSCD into TEST.

The IKJEGSUB macro that generates IKJEGSCD in MVS/370 is deleted from TSO/E for MVS/XA.

- UNALLOC is now an alias for the TSO FREE command. AND and OR are new subcommands of TEST. In type 32 SMF records, the UNALLOC command and the AND and OR subcommands of TSO TEST are recorded as *OTHER, unless your installation adds those names to CSECT IEEMB846.

TSO/E Considerations

When converting to MVS/XA, you may have to change certain TSO/E application programs. The changes you need depend upon the release of TSO/E to which you are converting. Consult the *TSO/Extensions (TSO/E) General Information* (GC28-1061) for details on the various TSO/E releases.

Deleted Instructions

The following instructions are deleted from the standard 370-XA instruction set:

ISK (Insert Storage Key)
SSK (Set Storage Key)
All 370 I/O instructions

Macro Expansions in JES Modifications

The MVS/XA expansions of 14 macros are downward incompatible. That is, the MVS/XA expansions will not work in MVS/370. As "Handling Downward Incompatible Macros" on page 9-7 explains, the MVS/XA MACLIB contains both the MVS/370 and MVS/XA expansions of those macros.

If you modify JES2 or JES3 system programs, you must take into account the level of macro expansions you need and how to obtain them. Depending on the JES you are working with, consult either *MVS/XA JES2 User Modifications and Macros* or *MVS/XA JES3 User Modifications and Macros* for relevant programming information.

Limiting Concurrent Global Resource Serialization Requests

Beginning with Release 2.1.1, global resource serialization limits the number of ENQ, RESERVE, and certain types of GQSCAN requests a single job, started task, or TSO user can have outstanding at a given time. The GQSCAN requests it limits are those that specify the TOKEN parameter. The change is designed to prevent one address space from using up all of GRS virtual storage, which causes subsequent GRS requests to fail.

Generally, the new processing does not require any action on your part. However, you need to be aware of the changes. Users might receive new ABEND or return codes indicating their programs failed because of too many concurrent global resource serialization requests. Also, you might want to change the limits global resource serialization enforces, although the default values are satisfactory for most installations.

To enforce the limit, as of Release 2.1.1 global resource serialization uses a threshold for each address space. The thresholds are in the GVTCREQ fields of the GVTs. If the number of outstanding ENQ, RESERVE, or specific types of GQSCAN requests reaches the threshold (the default is 4096), global resource serialization:

- Rejects subsequent ENQ and RESERVE requests from *unauthorized* callers in the address space. The system terminates unconditional requests with ABEND code x'538', and rejects conditional requests with a return code of x'014'. In earlier releases, if GRS virtual storage is depleted, users receive a return code of x'08'.

- Allows *authorized* callers in the address space to issue a limited number of additional ENQ and RESERVE requests. The number cannot exceed the tolerance value specified in the GVTCREQA field of the GVT. The tolerance value is also new as of Release 2.1.1. Its default value is 4111. Global resource serialization allows authorized callers the additional ENQ and RESERVE requests to enable recovery and normal termination routines to obtain the resources required to finish processing.

- Rejects with a return code of x'14' GQSCAN requests that specify the TOKEN option and request more information than can fit into the caller's buffers. Global resource serialization returns the buffers of information but does not continue the scan as it normally would. (If the threshold had not been reached, global resource serialization would have queued the request for continuation, returned the full buffers to the caller, and, after the caller cleared the buffers, resumed the scan.)

If you find the threshold and tolerance values in the GVT are too high or low, you can change them for your installation using the SPZAP service aid program. For details, see *SPL: Service Aids*.

Format Changes to Hard-Copy Log Records

Beginning with Release 2.1.2, the formats of all hard-copy log records except those written using JES3 are changed to provide additional machine-readable information. As a result, you need to modify most programs that scan the SYSLOG data set. Scan programs that run on a JES3 system might work unchanged. However, be aware that records logged before JES3 is initialized and all records written via the LOG command or the WTL macro have the new format. If your installation keeps the hard-copy log on a JES2 multi-access spool that systems at earlier levels can access, the scan programs must be sensitive to which system wrote the record.

The new log format includes the following additional information:

- A record ID, which identifies the type of record written (for example, a WTOR, label line, or command response). The record ID appears only in SYSLOG and not in printed output.

- The system ID.

- The date the message was issued.

- The ID of the console from which the command or command response was issued.
- User exit and message suppression flags.
- The full text of the message. If the text requires more than one line, one or more WTL entries might be interspersed among the continuation lines. If printed, however, the text appears on consecutive lines.

  The text of entries written using the LOG command begin with the prefix '0'. The text of entries written using the WTL macro begin with either a user-specified prefix or an 'X'.

A new macro, IHAHCLOG, maps the new record format. When modifying your programs, use the mapping macro instead of offsets to access the data.

Link Editing Allocation User Routines

Release 2.1.1 removes the following routines from the device allocation load module, IEFW21SD. Therefore, you need to link edit them differently:

IEFDB401 - Dynamic allocation user exit, now a single CSECT
IEFXVNSL - Non-standard tape label routine, now a single CSECT
IEFAB445 - Allocation space defaults CSECT, now a single CSECT
IKJEFD00 - Now an alias for the IKJDAIR dynamic allocation interface routine within the IEFGB400 load module.

IEFDB401 and IEFXVNSL can reside above or below 16 megabytes. Specify their RMODEs on the link edit statements for each. The system generation link edit control statements omit the RMODE specification. IEFAB445 resides below 16 megabytes. IKJDAIR also resides below 16 megabytes and can be invoked in either 24- or 31-bit addressing mode.

Release 2.1.1 changes some of the entry points in IEFW21SD. Programs to be executed in 31-bit addressing mode must use the new rather than old entry points. See "Entry Points in IEFW21SD" below for more information.

Entry Points in IEFW21SD

The following entry points in the device allocation load module, IEFW21SD, are changed in Release 2.1.1. When writing programs to be executed in 31-bit addressing mode, use the new entry points. Programs that run in 24-bit addressing mode can continue to use the old entry points.

Release 2.1.0 Entry Point	**Release 2.1.1 Entry Point**
IEFAB4DC	IEFGB4DC
IEFAB445	IEFAB445 (single CSECT)
IEFAB4UV	IEFGB4UV

Removal of the Interval Timer

370-XA deletes the interval timer. Modify programs that use the interval timer to use the CPU timer instead. Although the CPU timer works like a stopwatch, you can use it like an interval timer. Use the STIMER macro to set the CPU timer, the CPUTIMER or TTIMER macro to obtain its current value, and the SRBTIMER macro to set a time limit for SRB processing. Although you can use either the CPUTIMER or TTIMER macro, CPUTIMER is faster and you can use it in SRB or task mode. You can use TTIMER in task mode only.

Checklist for Determining if Authorized Programs Must be Changed

The following checklist is for use in examining **authorized** assembler programs for incompatibilities and is not applicable to other programs. Programs written in high level (non-assembler) languages are compatible and require no change. Most unauthorized assembler programs also work unchanged. The few exceptions are noted in the introduction to this chapter.

Most of the following programs require modification and/or reassembly:

- Programs that issue any of the following macros:

  RESETPL (a BTAM macro)
  IOHALT (or SVC 33)
  IOSGEN UCBLOOK
  STATUS STOP,SYNCH

  In all cases, you can change programs that use these macros before installing MVS/XA. For details, see the topics describing the macros.

- Programs that access system control blocks that are changed or that now reside in virtual storage above 16 megabytes. "Appendix B. Control Block Changes" lists the control blocks requiring attention.

- Programs that directly invoke system modules that now require entry in 31-bit addressing mode, or that require parameter addresses to be 31-bit values.

  Programs using SVCs or published macros to invoke service routines that now execute in 31-bit addressing mode generally work unchanged in MVS/XA. In most cases, the macro invokes a routine that changes modes, if necessary, before entering the service routine.

  The MVS/XA components having a large percentage of modules that execute in 31-bit addressing mode or reside above 16 megabytes include:

  - In Release 2.1.0

    - BLDL/FIND
    - checkpoint/restart (runs in 31-bit addressing mode)
    - contents supervision (CSV)
    - some device error recovery procedures (ERPs)
    - GTF
    - IOS

  - media manager
  - program fetch
  - RSM
  - RTM
  - SRM
  - system trace
  - VSM

- In Release 2.1.1
  - ALLOCATE

- In Release 2.1.2
  - SVCDUMP in APAR 0Z78216
  - VSAM record management load modules

- In Release 2.1.3AE
  - availability manager

- In MVS/XA DFP Version 2 Release 1.0
  - checkpoint/restart (resides above the 16 megabytes address)
  - common volume table of contents access facility (CVAF)

- In MVS/XA DFP Version 2 Release 3.0
  - VSAM

- RMF Version 3 modules also execute in 31-bit addressing mode.

See "Interfaces to System Services" for more detail.

- Programs that use the high-order byte of address fields for flags. When running in 31-bit addressing mode, MVS/XA treats addresses as 31-bit values and, if applicable, uses the high-order bit to set the PSW A-mode bit.

  Specific examples of programs that will fail include those that use the high-order byte of:

  - Address fields they pass to IARUTRV (translate real to virtual routine). IARUTRV, which replaces IEAVTRV in MVS/XA, treats the real addresses as 31-bit values.

  - The SRBEP or SRBRMTR field in the SRB. MVS/XA treats each field as a 31-bit value, and uses the high-order bit to set the PSW A-mode bit.

  - The SVC screening table address in the TCB (the TCBSVCA2 field). MVS/XA also treats that address as a 31-bit value and uses the high-order bit to set the PSW A-mode bit.

- Programs that treat UCB addresses as 2-byte values. In MVS/XA, UCB addresses are three bytes instead of two.

- Programs that directly access the UCB look-up routine. It does not exist in MVS/XA. "IOSGEN UCBLOOK Macro Instruction" describes alternate ways of obtaining the same information.

- Programs that depend on the structure of the nucleus and the FLPA. In MVS/XA, the FLPA no longer resides in the nucleus buffer. Also, neither the nucleus nor the FLPA is mapped V=R, and modules in those areas might not be loaded into contiguous real frames.

  Examples of programs that must be modified are:

  - V=R programs that use EXCP to perform I/O into or out of the FLPA.

  - Programs in the nucleus or FLPA that run DAT-off. "DAT-off Restrictions" describes how to change the programs.

  - Programs that use the CVTNUCB field to determine if they have been loaded into the FLPA. Change the programs to test the CVTFLPAS and CVTFLPAE fields to determine residency in the FLPA below 16 megabytes and the CVTEFLPS and CVTEFLPE fields for residency above 16 megabytes. Each pair of fields indicates the beginning and ending addresses of the FLPA areas.

- Programs sensitive to virtual storage location changes; for example, programs that treat parameter addresses below 64 K as invalid.

- Programs sensitive to changes in the locking structure. Programs requiring modification are those that:

  - Use the IOSCAT or the IOSLCH lock. Those locks are deleted in MVS/XA.

  - Obtain the DISP lock in order to hold the highest lock in the system.

  - Request the DISP lock after obtaining the ASM lock.

  - Use the PSAHLHI field to determine locking hierarchy.

  For more information, see "Changes to the Locking Structure" and "Determining Which Locks a Processor Holds."

- Programs that use the following system-created data:

  - GTF, system trace, or LOGREC records. The record formats are different. Also, the structure of the system trace table is changed.

  - Beginning in Release 2.1.2 programs that scan the SYSLOG data set must be updated as a result of changes in the format of the hard-copy log records. See "Format Changes to Hard-Copy Log Records."

  - Dump data. Dump contents and formats have changed. See Chapter 6 for more information.

- SMF data that is updated. Chapter 7 identifies which SMF records are updated and briefly describes the differences.

- Programs that examine the PSW field in MVS/XA control blocks (for example, programs that use trace data or print reports). The PSW format is changed. Among other differences, the instruction addresses are contained in 4-byte, instead of 3-byte, fields.

- Programs that use the LRA instruction. LRA always returns a 31-bit address in MVS/XA, even when executed in 24-bit addressing mode.

- Programs that call IEFSCAN or that directly access MVS/370 device allocation tables (DEVNAMET, IEFDEVPT, and DEVMASKT). IEFSCAN and the tables are deleted in MVS/XA. "Unit Verification" describes how both authorized and unauthorized programs can perform unit verification in MVS/XA.

- Programs that use extended ECBs for POST exits. The programs must be authorized to fetch from the ECB extension, as well as to fetch and store the extended ECB.

- Programs that specify an ACON length other than 4 (for example, AL3(location)), if the location in parentheses is above 16 megabytes.

- Programs that examine the SVTDACTV or SVTPWAIT fields in the SVT (usually programs that code their own expansions of SCHEDULE or INTSECT, respectively). The offsets of these fields in the SVT have changed. Their previous locations are initialized to x'FFFFFFFF'.

- Programs that depend on CPU (processor) addresses being 0, 1, or 2. A CPU address can be any number from 0-F.

- I/O drivers that call IEASMFEX to record EXCP counts. Change the drivers to use a new SMF macro, SMFIOCNT.

- If your installation includes data sets in the LNKLST concatenation that are not APF authorized, programs that depend on the data sets being APF authorized.

  The DEBAPFIN bit in the LNKLST DEB indicates whether or not all data sets in the LNKLST concatenation are APF authorized. The LLTAPFIN field in a data set's LLTAPFTB entry indicates whether the data set is APF authorized. The LLTAPFTB is a new extension to the LLT that contains one entry for each data set in the LNKLST concatenation. See "Using a New Directory for LNKLST Data Sets" in Chapter 8, "Measurement and Tuning" for more information.

- Programs that access SMF BQEs (buffer queue elements). Release 1.1 moves the BQEs from common storage to the new SMF address space.

- Programs that read SMF data sets directly instead of via SMF dump programs. Release 2.1.1 initializes SMF data sets with dummy records that are shorter than valid SMF records. They contain the characters 'SMFEOFMARK.' See "SMF Compatibility Between Release 2.1.0 and Later Releases" in Chapter 7 for more information.

- Programs that obtain storage for data extent blocks (DEBs) from fetch-protected areas (subpools 0-172). If called to add a DEB table entry for a DEB that is in fetch-protected storage, the MVS/XA DEBCHK service routine issues ABEND x'16E' with reason code x'1C'. MVS/370 does not impose the same restriction.

Changes to the SVC Table

The following changes have been made to the SVC table:

SVC	Description of Change
SVC 16 (PURGE)	Changed from type 3 to type 2
SVC 46 (TTIMER)	Changed from type 3 to type 2
SVC 47 (STIMER)	Changed from type 3 to type 2
SVC 82 (DASDR)	Deleted
SVC 88 (MOD88)	Deleted
SVC 109 (Extended SVC Router)	A new entry has been added: 28 - ESPIE, a type 3 SVC
SVC 138 (PGSER)	A new type 2 SVC

Changes to the Locking Structure

The locking structure has changed in MVS/XA:

- MVS/XA uses nine new locks instead of the SALLOC lock for storage management serialization.
- A new TRACE lock serializes the system trace buffer structure.
- A new CPU lock causes the requestor to be physically disabled for I/O and external interrupts. It provides system-recognized disablement.
- The IOSCAT and IOSLCH locks have been deleted.
- The hierarchy of the ASM and DISP locks is reversed. In MVS/XA, the ASM lock's position is above the DISP lock's position in the locking hierarchy.

You must change programs that:

- Use the IOSCAT or IOSLCH locks.
- Use the SALLOC lock to serialize storage management.
- Obtain the DISP lock in order to hold the highest lock in the system.
- Request the DISP lock after obtaining the ASM lock.

Determining Which Locks a Processor Holds

You must change programs that use the PSAHLHI (highest lock held) field to determine locking hierarchy. The PSAHLHI field is now referred to as PSACLHS (current locks held string), although the old name is retained for compatibility. The bit positions in the PSACLHS field indicate which locks the processor owns. They no longer represent the hierarchy of locks.

MVS/XA provides a new SETLOCK service that indicates whether the processor owns any locks at a higher position in the hierarchy than the one specified as input. For example, the following SETLOCK macro tests whether the processor owns any locks at a higher position than the dispatcher lock:

```
SETLOCK TEST,TYPE=HIER,LOCK=DISP,REGS=(11,12)
```

Page Protection

MVS/XA uses a new page-protection facility to enforce read-only access to the:

- Read-only nucleus (above and below 16 megabytes)
- Resident BLDL list in Release 2.1.0
- PLPA (above and below 16 megabytes)
- MLPA (above and below 16 megabytes)
- FLPA (above and below 16 megabytes)
- NUCMAP (an area in the non-page-protected nucleus that maps the nucleus)

Page protection is optional only for the MLPA or FLPA. Installations can turn off page protection for those areas by specifying a new subparameter, NOPROT, on the MLPA and FIX system parameters, respectively, in the IEASYSxx PARMLIB member. When NOPROT is specified, none of the MLPA or FLPA is page-protected. The system default is to page protect those areas.

You cannot include in page-protected areas any module that stores into itself. If any program running DAT-on attempts to store into a page that is page-protected, the processor generates a program interrupt, regardless of the program's state or protect key.

Modules that modify themselves might include:

- Modules that use macros which create parameter lists, but do not use the LIST/EXECUTE forms of the macros to eliminate stores into the module
- Modules marked reentrant that page-fix, serialize, and modify themselves

If you have any such modules in the PLPA, either:

1. Modify the module so that it stores the data somewhere else; for example, in dynamically-acquired storage.
2. Include the module in another library; for example, in SYS1.LINKLIB.
3. Include the module in the MLPA or FLPA and specify NOPROT for that area.

Unless you turn off page protection in the MLPA or FLPA, handle self-modifying modules in those areas as described in 1 or 2.

The page protect facility replaces MVS/370 segment protection, which enforces read-only access to segments (64 K blocks) of storage fully occupied by PLPA pages. MVS/370 installations can override the segment protection using the AMASPZAP service aid program. Installations do not have that capability in MVS/XA.

PSA Low Address Protection

PSA low address protection prevents user programs from storing into PSA locations 0 through 511. The only way to turn off PSA low address protection in MVS/XA is by using the PROTPSA macro. The CVTPRON bit in the CVT is deleted.

To disable low address protection in MVS/370, programs can either use the PROTPSA macro or change the CVTPRON bit.

Fetch-Protected PSA Areas

You might have to change some programs that fetch data from the PSA. In MVS/XA, some PSA locations are key 0 fetch-protected. That is, only programs in key 0 can fetch data from those areas:

- The last 2 K of the PSA (2 K through 4 K minus 1) are always key 0 fetch-protected.
- The first 2 K are key 0 fetch-protected from programs running on a different central processor. However, programs require no authorization to fetch data from the first 2 K of the PSA of the central processor on which they are running. The programs must still be in key 0 to store data into the first 2 K, however.

The PSA work/save areas have been moved into the fetch-protected area to improve integrity. Therefore, you must also modify programs that access the moved data while not in key 0 and programs (usually FRRs) that fetch data from the FRR six-word parameter area while not in key 0.

Fetch-protection of PSA locations is new in MVS/XA. In MVS/370, programs have to be in key 0 to store into the PSA, but no authorization is required to reference data there.

Patch Areas in the PSA

MVS/XA uses some PSA locations that are available for system patches in MVS/370. If you use areas of the PSA for system patches, ensure that your patch applications use only areas that are not system-defined. Otherwise, normal system processing might overlay the patch.

To determine which areas are safe to use, patch applications can check whether the storage contains zeros. When initializing the PSA, MVS/XA puts non-zero values in the system-defined areas within the range most commonly used as a patch area (x'600' to x'C00'). Available areas contain zeros.

Real Addressing Considerations

You might need to change programs that:

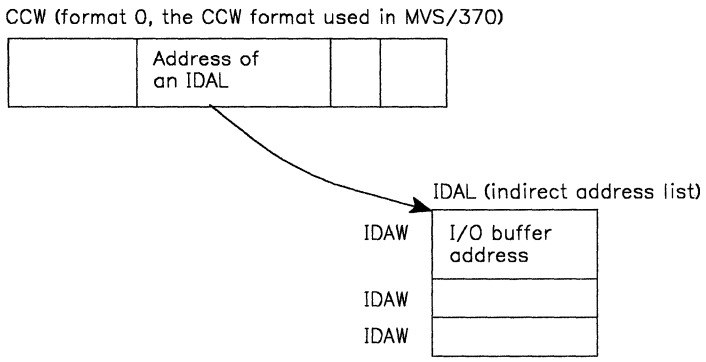
- Use the EXCPVR macro instruction
- Depend on RSM backing virtual pages with real storage below 16 megabytes
- Execute DAT-off code

Using the EXCPVR Macro Instruction

EXCPVR users need to be aware of two changes:

- The list of data areas that the page fix (PGFX) appendage passes to the I/O supervisor must contain 31-bit addresses. The high-order bit of each address must be zero.

- Using EXCPVR, CCWs (channel command words), and IDAWs (indirect address words), programmers can perform I/O to any location in real storage (above or below 16 megabytes). The channel programs must use IDAWs to specify the address of buffers in real storage above 16 megabytes



Because the EXCP service routine (which processes both EXCP and EXCPVR macros) supports only Format 0 CCWs, CCWs and IDAWs used with EXCPVR must reside in virtual storage below 16 megabytes.

As in all cases where IDAWs are used with real addresses, an IDAW is required for each 2 K real storage boundary that the data transfer operation will cross.

Unless a program uses IDAWs, EXCPVR users must ensure that all buffers are backed by real storage below 16 megabytes. EXCPVR users must assume that buffers obtained via data management access methods have real addresses above 16 megabytes because data management access methods specify LOC=(BELOW,ANY) on the GETMAIN request for buffer storage. LOC=(BELOW,ANY) indicates that virtual storage can be backed with real storage above 16 megabytes. (See "Parameters on the GETMAIN Macro Instruction.") The following situation can happen:

1. A program uses an access method to open a data set. The access method obtains buffer storage that might be backed by real storage above 16 megabytes.

2. The program uses the access method to read data into the buffer.

3. The program attempts to write data from the buffer using EXCPVR. If the buffer is in real storage above 16 megabytes, the program does not work unless it uses IDAWs to specify the real addresses above 16 megabytes.

Change the program to either obtain its own buffer or use IDAWs.

MVS/Extended Architecture System Programming Library: System Data Administration describes how to use EXCPVR.

Changes in the Way RSM Backs Virtual Storage

RSM uses different algorithms to determine whether to back a virtual page with real storage above or below 16 megabytes. Generally, only users who have programs with real address dependencies need to be aware of the changes. RSM:

- Attempts to back all virtual storage **above** 16 megabytes with real storage **above** 16 megabytes.

- Attempts to back the following virtual storage areas **below** 16 megabytes with real storage **above** 16 megabytes:

  SQA (except subpool 226)
  LSQA
  Nucleus
  Pageable private areas
  Pageable CSA
  PLPA
  MLPA
  Resident BLDL (in Release 2.1.0 only)

- Always backs the following virtual storage areas **below** 16 megabytes with real storage **below** 16 megabytes:

  V=R regions
  FLPA
  Subpool 226 (a new subpool in SQA)

- Backs subpools 227 and 228 (fixed CSA) in virtual storage **below** 16 megabytes with real storage **below** 16 megabytes, except when GETMAIN requests specify LOC=(BELOW,ANY).

- When satisfying a page-fix request, RSM generally backs pageable virtual pages that reside **below** 16 megabytes with real storage **below** 16 megabytes. (Pageable virtual pages are pages in CSA, PLPA, MLPA, or the pageable private area.) However, in the following situations, RSM attempts to use real storage **above** 16 megabytes:

  - The GETMAIN request to obtain the storage specified either LOC=(BELOW,ANY), LOC=(RES,ANY), or LOC=(ANY,ANY).

  - The PGSER macro specified ANYWHERE

Note: EXCPVR users need to be aware that MVS/XA DFP access methods use LOC=(BELOW,ANY) on GETMAIN requests for buffer storage.

Impact on Programmers:

RSM's page backing rules are, for the most part, compatible with the way real storage is backed in MVS/370. Because programs that have real address dependencies work with fixed storage, it is expected that most existing programs will continue to receive real addresses that are less than 16 megabytes. However, you must change programs that run in 24-bit addressing mode and have real address dependencies on the nucleus, SQA, or LSQA. RSM ignores requests to fix storage in the nucleus, SQA, or LSQA because those areas are already fixed. Therefore, real addresses in those areas might be greater than 16 megabytes. Modify the programs to correctly handle 31-bit addresses.

DAT-off Restrictions

You must modify programs in the nucleus or FLPA that run with dynamic address translation (DAT) turned off. In MVS/370, programs can turn DAT on or off by manipulating the system mask (using the STNSM and STOSM instructions). However, because the nucleus and FLPA are not mapped V=R in MVS/XA, modules in those areas can no longer use the STNSM and STOSM instructions to control DAT. (Programs executing as V=R jobs can use STNSM and STOSM instructions in MVS/XA and do not have to be modified unless they refer to data outside the V=R region.)

To modify modules containing DAT-off code:

1. Move the DAT-off code to a separate module. Give the module AMODE=31 and RMODE=ANY attributes. Use as its entry point, IEAVEURn, where n is a number from 1 to 4. (MVS/XA reserves four entry points in the DAT-off nucleus for users.) Use BSM 0,14 as the return instruction.

2. In the original module (which executes DAT-on), code a DATOFF macro to invoke the DAT-off module created in the previous step. DATOFF is new in MVS/XA:

   ```
   DATOFF INDEX=INDUSRn
   ```

   The suffix of INDUSRn must be the same as the suffix of the DAT-off module's entry point, IEAVEURn. See *System Macros and Facilities* for more detail on coding DATOFF macros.

3. Link edit the DAT-off module (IEAVEURn) into the IEAVEDAT member of SYS1.NUCLEUS (the DAT-off nucleus).

When the DATOFF macro executes, it branches to a routine in the PSA. The routine turns DAT off and branches to entry point IEAVEURn in IEAVEDAT. The DAT-off module returns via a PSA routine that turns DAT back on.

Cross-Memory Entry Table Entries

You might have to change entry table entries that your installation created. MVS/XA uses a previously reserved bit in cross-memory entry table entries to determine the addressing mode in which to enter the program. Entries that require modification are those that specify program addresses and either:

- Use bits in the entry description that are reserved in MVS/370
- Specify programs to be entered in 31-bit addressing mode.

Interfaces to System Services

Some system services are changed to execute in 31-bit addressing mode. Some can now accept callers in either mode, but have restrictions on the length or value of parameter addresses. Others are restricted to using MVS/370-supported interfaces. During the migration phase, most programmers do not have to be concerned about changes to system service interfaces. Most programs that execute in 24-bit addressing mode and invoke system services via an SVC or a macro instruction continue to work unchanged in MVS/XA. (Exceptions are noted in the introduction to this chapter.) Interface changes might, however, affect existing programs that invoke service routines directly instead of via an SVC or macro instruction.

When modifying or developing programs that invoke system services directly or that execute in 31-bit addressing mode, programmers must now consider:

- The mode of the caller.
- The desired mode of the routine being called.
- The location of data areas passed to the service routines. Some data areas, such as the DCB, cannot reside in virtual storage above 16 megabytes.
- The location of routines whose addresses are passed as parameters.
- The length of the address parameter fields. Some services expect parameter address fields to be 31 bits long even though the addresses contained in the fields might point to locations in virtual storage below 16 megabytes. Other services use parameter fields that must be 24 bits long (for example the DCB address in an OPEN parameter list).

Programmers need to refer to the publications documenting the macros and SVC interfaces when using system services in 31-bit addressing mode or when invoking them directly.

System services can be categorized according to their interface requirements. Following are descriptions of the categories and examples of service routines in each. The list is not comprehensive.

Services Independent of Addressing Mode

Service routines in this category:

- Accept callers in either 24- or 31-bit addressing mode.
- Use 31-bit parameter address fields or the 31-bit MODE parameter and, for callers in 31-bit mode, allow the addresses contained in those fields to point to any location.

EXAMPLES:

ABEND	EVENTS	PUT*
ATTACH**	EXIT	RESERVE
BLDVRP*	FESTAE	RESTORE
CALLRTM	FREEMAIN (SVC 120)	SCHEDULE
CHAP	GENCB*	SDUMP
CIRB	GET*	SETFRR
CLOSE	GETMAIN (SVC 120)	SETLOCK
CMSET	GETSRB	SETRP
CPOOL	GTRACE	SNAP**
DATOFF	GQSCAN	STATUS
DELETE	HOOK	STIMER
DEQ	IDENTIFY	SYNCH
DETACH	LINK**	SYSEVENT
DLVRP*	LOAD**	TESTAUTH
DOM	MODCB*	TESTCB*
DYNALLOC	OPEN	TTIMER
ENQ	PGSER	WAIT
ESPIE	POST	WTO
ESTAE	PTRACE	WTOR
		XTCL**

**When a DCB parameter is specified, the DCB must reside in 24-bit addressable storage.

*VSAM macros

Services with Some Restrictions on the Address Parameter Values

Services in this category:

- Accept SVC callers in either 24- or 31-bit addressing mode.
- Might require that branch entry callers be in 24-bit addressing mode.
- Require that one or more parameter addresses point to locations below 16 megabytes. In some cases, the length of an address field must be 24 bits. In other cases, the length of an address field must be 31 bits long, but the address contained in the field must be a 24-bit value.

EXAMPLES:

BLDCPOOL	GETMAIN (SVC 4 and 10)	RACDEF
EVENTS	MGCR	RACLIST
EXCP	PGFIX	RACHECK
EXCPVR	PGFREE	RACINIT
EXTRACT	PGLOAD	RACROUTE
FRACHECK	PGOUT	SHOWCM
FREECELL	PGRLSE	SMFWTM
FREEMAIN (SVC 5 and 10)	PURGE	SMFEWTM
GETLINE	PUTLINE	STACK
GETCELL	QEDIT	

Note: The fullword value passed as an address must be appropriate to its subsequent use by the system service.

Services that Do Not Support 31-bit Addressing

Services in this category:

- Accept callers in 24-bit addressing mode only.
- Require that all parameter addresses point to storage below 16 megabytes. Parameter lists (both in-line and remote), control blocks, buffers, and user exit routines must reside in virtual storage below 16 megabytes.

EXAMPLES:

SPIE
STAE
SEGLD
SEGWT

Data management macro instructions for all DFP access methods except VSAM (specifically, SAM, PAM, DAM, and ISAM)

31-bit Addressing Considerations

A 370-XA system can treat instruction and data addresses as 24- or 31-bit values. A new concept, addressing mode, describes the size of addresses being used. The value of a bit in the PSW (the PSW A-mode bit) determines the addressing mode. If the bit is 0, the system treats addresses (except those returned from the LRA instruction) as 24-bit values. If the bit is 1, the system treats them as 31-bit values. Programs executing while the system is in 24-bit addressing mode can address up to 16 megabytes of virtual storage. Programs executing in 31-bit mode can address up to two gigabytes (approximately 2 billion bytes) of virtual storage.

Impact of 31-bit Addressing on Programmers

During the migration phase, most programmers do not have to be concerned with addressing mode. Most existing user-written programs that use standard system interfaces run unchanged on an MVS/XA system in 24-bit mode.

Programmers need to be concerned about addressing mode only if they:

- Have existing user-written programs that access system control blocks that have been moved to virtual storage above 16 megabytes. (Appendix B, "Control Block Updates" lists those control blocks.) Programs that run in 24-bit addressing mode must switch modes to access data above 16 megabytes. The next topic describes ways of changing the addressing mode. "Retrieving Data from a Control Block Above 16 Megabytes" illustrates how 24-bit mode programs can be changed to reference virtual storage above 16 megabytes.

- Have existing user-written programs that use non-standard interfaces to invoke system programs (for example, programs that branch enter system programs rather than use macros, SVCs, or documented entry points). Some system programs must now be entered in 31-bit addressing mode or using a BASSM instruction.

  Also, some system programs now expect input addresses to be 31-bit values. Modules that run in 24-bit mode must ensure that the addresses they pass to programs in 31-bit mode do not contain flags or other data in the high-order byte, unless the 31-bit mode program ignores the first byte or sets it to zero.

  See "Modifying Programs that Invoke Modules Above 16 Megabytes" for examples of how you can make affected programs work in MVS/XA.

- Develop application programs, exit routines, or system modifications that execute in 31-bit addressing mode. Developing new programs to execute in 31-bit addressing mode is not described in this publication. See *SPL: 31-Bit Addressing*.

The following address mode related topics give programmers an introduction to how mode setting is performed so they can assess the work required to modify existing programs:

- "Changing Addressing Mode"
- "Establishing a Program's Addressing Mode"
- "BSM (Branch and Set Mode) Instruction"
- "BASSM (Branch and Save and Set Mode) Instruction"
- "Modifying Programs that Invoke Modules Above 16 Megabytes"
- "Retrieving Data from a Control Block Above 16 Megabytes"
- "Performing I/O in 31-bit Addressing Mode"
- "Using the EXCP Macro"

See *SPL: 31-bit Addressing* for more detail.

The only way to change the addressing mode is to change the value of the PSW A-mode bit. Following are ways of changing the A-mode bit:

- New 370-XA instructions:

  BSM (branch and set mode)
  BASSM (branch and save and set mode)

  Both BSM and BASSM can save the current addressing mode, set a new addressing mode, and branch to an address. BASSM also saves a return address. The instructions allow problem programs in different addressing modes to communicate. See "New Instructions" for more detail.

- Supervisor assisted linkages (XCTL, LINK, and ATTACH). When a module uses XCTL, LINK, or ATTACH to invoke another routine, MVS/XA ensures that the called routine receives control in the correct addressing mode. (The way programs establish an addressing mode is described in the next topic.) Programs issuing XCTL, LINK, or ATTACH macros do not have to be aware of the addressing mode of the called routines except to ensure that the parameter requirements are met. When the routine called using LINK or ATTACH returns, the supervisor restores the addressing mode of the caller.

- Supervisor calls (SVCs). The supervisor saves and restores the issuer's addressing mode and ensures that the service routine receives control in the correct mode.

  Programs that reside below 16 megabytes and pass parameters located in virtual storage below 16 megabytes can issue SVCs without being aware of the service routine's addressing mode or input requirements. However, before using SVCs in programs that will execute in 31-bit mode and/or use parameters located above 16 megabytes, consult documentation on the SVC interface. Some SVCs require that input parameters be located below 16 megabytes. See "Interfaces to System Services" for more detail.

- SYNCH macro. A new parameter, AMODE, allows programs to specify the addressing mode in which the called routine is to get control.

- SRB dispatch. When the SRB is dispatched, MVS/XA replaces the PSW A-mode bit with the high-order bit of the SRBEP or SRBRMTR field.

- PC and PT instructions, which establish the identified addressing mode.

- LPSW instruction.

Establishing a Program's Addressing Mode

Every program that executes in MVS/XA is assigned two new attributes, an AMODE (addressing mode) and an RMODE (residency mode). (Existing programs are assigned default AMODE/RMODE attributes, which are described below.) AMODE specifies the addressing mode in which the program is designed to receive control. Generally, the program is also designed to execute in that mode, although a program can switch modes and can have different AMODE attributes for different entry points within a load module. The RMODE indicates where in virtual storage the program can reside.

Valid AMODE and RMODE specifications are:

AMODE=24	Specifies 24-bit addressing mode
AMODE=31	Specifies 31-bit addressing mode
AMODE=ANY	Specifies either 24- or 31-bit addressing mode
RMODE=24	Indicates that the module must reside in virtual storage below 16 megabytes. You can use the RMODE=24 specification for 31-bit programs that have 24-bit dependencies.
RMODE=ANY	Indicates that the module can reside anywhere in virtual storage

You do not have to specify AMODE and RMODE attributes for a program. When none are specified, the system assigns the following defaults: AMODE=24, RMODE=24. To override the defaults, specify AMODE and/or RMODE on one or more of the following:

- AMODE and RMODE statements within the assembler source code for a program. Only Assembler H Version 2 recognizes AMODE and RMODE statements.

```
XYZ     CSECT
XYZ     AMODE xxx
XYZ     RMODE xxx
```

- The EXEC statement of a linkage editor step:

```
//LKED    EXEC    PGM=HEWLH096,PARM='AMODE=xxx,RMODE=xxx, ...'
```

- The LINK TSO command:

```
LINK    AMODE(xxx),RMODE(xxx)
```

  Giving this command causes processing equivalent to that of the EXEC statement on a linkage editor step.

- The EXEC statement of a loader step:

```
//LOAD    EXEC    PGM=LOADER,PARM='AMODE=xxx,RMODE=xxx, ...'
```

- The LOADGO TSO command:

```
LOADGO AMODE(xxx),RMODE(xxx)
```

  Giving this command causes processing equivalent to that of the EXEC statement on a loader step.

- The linkage editor MODE control statement (one per load module):

```
MODE    AMODE(xxx),RMODE(xxx)
```

AMODE/RMODE specifications given in EXEC statements or TSO commands override specifications in the program code. AMODE/RMODE specifications in the linkage editor MODE control statement override specifications in the linkage editor EXEC statement, the TSO LINK command, and the program code.

MVS/XA uses a program's AMODE attribute to determine whether a program invoked using ATTACH, LINK, or XCTL is to receive control in 24- or 31-bit addressing mode. MVS/XA uses the RMODE attribute to determine whether a program must be loaded into virtual storage below 16 megabytes or can reside anywhere in virtual storage (above or below 16 megabytes).

Assembler H Version 2 establishes flags in the external symbol dictionary (ESD) to indicate the specified (or default) AMODE and RMODE of each CSECT. The MVS/XA linkage editor retains these flags in the composite external symbol dictionary (CESD). The linkage editor also inserts AMODE and RMODE flags in the partitioned data set (PDS) directory entry for each load module. The linkage editor by default uses the AMODE and RMODE indicators from the ESD entries. As noted earlier, the linkage editor also accepts AMODE and RMODE specifications in the EXEC statement and in the MODE control statement. If either of these is used to specify AMODE or RMODE, they are reflected in the PDS directory entry, but do not affect the information in the CESD.

You can use the MVS/XA version of AMBLIST to print the directory entry and the CESD to determine a program's AMODE and RMODE. You can also use the LOAD macro to determine the addressing mode in which a module expects to receive control. The high order bit of the entry point address that LOAD returns indicates the addressing mode.

Note: Do not confuse AMODE with the current addressing mode. Specifying an AMODE attribute guarantees that the module will receive control in the specified mode **only** when invoked using one of the methods defined in this topic. Specifying an AMODE does not, for example, prevent a program in 24-bit addressing mode from issuing a BALR to a program with an AMODE of 31, although the program may not execute as expected. Also, there is nothing to prevent a programmer from specifying an AMODE of 31 on a program designed to execute in 24-bit mode, although doing so is incorrect.

Linkage Editor Interpretation of AMODE=ANY,RMODE=ANY

With MVS/370 DFP and MVS/XA DFP Version 1, the linkage editor interprets load module entry points with external symbols marked AMODE=ANY and RMODE=ANY as AMODE=31, RMODE=ANY. With MVS/XA DFP Version 2, the way the linkage editor interprets AMODE=ANY, RMODE=ANY specifications depends upon the characteristics of the load module. There are three possibilities:

1. If the load module contains at least one CSECT marked AMODE=24, the linkage editor interprets AMODE=ANY,RMODE=ANY as AMODE=24, RMODE=24.

2. If the load module contains no CSECTS marked AMODE=24 and the load module RMODE is 24, the linkage editor interprets AMODE=ANY, RMODE=ANY as AMODE=ANY, RMODE=24.

3. If the load contains no CSECTS marked AMODE=24 and the load module is RMODE=ANY, the linkage editor interprets AMODE=ANY, RMODE=ANY as AMODE=31, RMODE=ANY.

You can incorporate this MVS/XA DFP Version 2 linkage editor function into the MVS/370 DFP linkage editor by means of APAR OZ83849 (PTF UZ78097). You can incorporate this function into the MVS/XA DFP Version 1 linkage editor by means of APAR OZ82513 (PTF UZ78096).

Restrictions on Using a Linkage Editor Overlay Structure

Programs executing in 31-bit addressing mode cannot use a linkage editor overlay structure.

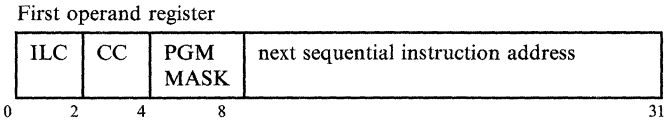
Changed Instructions

The following instructions work differently either when executed in 31-bit addressing mode or when executed on a 370-XA processor: BAL, BALR, BAS, BASR, CLCL, EDMK, LA, LRA, MVCL, and TRT. The following topics describe the differences.

Also remember that when executing in 31-bit addressing mode, 370-XA processors treat all virtual addresses as 31-bit values. When executing in 24-bit addressing mode, they treat virtual addresses as 24-bit values.

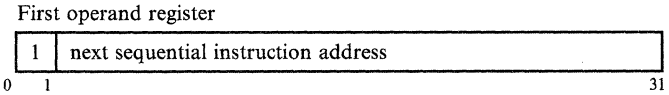
BAL and BALR (Branch and Link) Instructions

The way BAL and BALR work depends on the addressing mode. In 24-bit addressing mode, BAL and BALR work the same way as they do when executed on a 370 processor. BAL and BALR put information from the PSW into the high-order byte of the first operand register and put the return address into the remaining 3 bytes before branching:



ILC - instruction length code
CC - condition code
PGM MASK - program mask

In 31-bit addressing mode, BAL and BALR put the return address into bits 1 through 31 of the first operand and save the current addressing mode in the high-order bit. Because the addressing mode is 31-bit, the high-order bit is always a 1.



Note that when executed in 31-bit addressing mode, BAL and BALR do not save the instruction length code, the condition code, or the program mask. A new 370-XA instruction, IPM (INSERT PROGRAM MASK), saves the program mask and the condition code.

BAS and BASR (Branch and Save) Instructions

BAS and BASR instructions execute in either System/370 or 370-XA mode. They:

- Save the return address and the current addressing mode in the first operand.
- Replace the PSW instruction address with the branch address.

The high-order bit of the return register indicates the addressing mode.

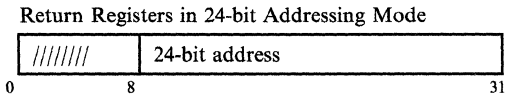
Note that BAS and BASR perform the same function that BAL and BALR perform when BAL and BALR execute in 31-bit addressing mode.

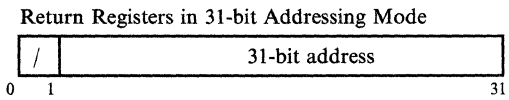
CLCL, EDMK, MVCL, and TRT Instructions

When executed in 31-bit addressing mode, the following four instructions load 31-bit values into return registers and leave bit 0 unchanged. When executed in 24-bit addressing mode, they load 24-bit addresses and leave bits 0-7 unchanged:

CLCL (Compare Logical Long)
EDMK (Edit And Mark)
MVCL (Move Long)
TRT (Translate And Test)

Most programs using these instructions will run unchanged in 31-bit addressing mode. You need to change only programs that depend on bits 1-7 remaining unchanged.





LA (Load Address) Instruction

The LA instruction works differently when executed in 31-bit addressing mode. It loads a 31-bit value and clears the high-order bit instead of the entire high-order byte.

LRA (Load Real Address) Instruction

The LRA instruction performs the same functions as in System/370 mode. However, it always returns a 31-bit address regardless of the issuing program's addressing mode. Also, the meaning of condition code 1 (CC=1) from an LRA instruction might be different. Because some page tables for the user region above 16 megabytes are themselves pageable in MVS/XA, a condition code of 1 can mean either that:

- The page table does not exist because the virtual space has not been obtained.
- The page table is paged out or has not yet been built.

In MVS/XA, neither situation is necessarily an error. Users issuing a page-fix before the LRA avoid the first possibility. For them, condition code X'01' means the page table is paged out or not yet built. Users that do not page-fix and receive a condition code of 1 can access the page in question to determine which condition exists. Accessing the page causes either an x'0C4' program check or segment fault/page fault resolution.

New Instructions

Standard instructions that are new in 370-XA mode include:

BASSM (BRANCH AND SAVE AND SET MODE)
BSM (BRANCH AND SET MODE)
CFC (COMPARE AND FORM CODEWORD)
DXR (DIVIDE, extended operand)
IPM (INSERT PROGRAM MASK)
TRACE
UPT (UPDATE TREE)
All I/O instructions

The following topics describe BSM and BASSM in more detail. For more information on the other instructions see *Principles of Operation.*

As of Release 2.13VFE, there are 63 vector instructions with 171 operation codes. These are documented in *Vector Operations.*

BSM (Branch and Set Mode) Instruction

BSM is a branch instruction that also sets the addressing mode.

The BSM instruction:

- Saves the current addressing mode. BSM puts into the high-order bit of the first operand the value of the current PSW A-mode bit. The rest of the operand is unchanged.

- Sets a new addressing mode. BSM replaces the PSW A-mode bit with the high-order bit of the second operand.

- Replaces the PSW instruction address with the branch address. Note that BSM sets the new addressing mode before computing the branch address. Thus, a program executing in 24-bit addressing mode can use BSM to branch to a program in virtual storage above 16 megabytes.

Uses for BSM

Programs called via a BASSM instruction (described in the next topic) can use BSM to return to the caller in the caller's addressing mode.

When the first operand is 0 (for example, BSM 0,14), BSM:

- Does NOT save the current addressing mode
- Sets the PSW AMODE bit
- Executes a branch

When the second operand is 0 (for example, BSM 14,0), BSM:

- Saves the current addressing mode
- Does NOT change the PSW AMODE bit
- Does NOT execute a branch

BASSM (Branch and Save and Set Mode) Instruction

The BASSM instruction works like BSM, except that it saves the return address as well as the current addressing mode in the first operand.

BASSM:

- Saves the next instruction address in bits 1 through 31 of the first operand.
- Saves the current addressing mode in the high-order bit of the first operand.
- Replaces the PSW A-mode bit with the high-order bit of the second operand.
- Replaces the PSW instruction address with a branch address. Like BSM, BASSM sets the new addressing mode before computing the branch address. Thus, programs executing in 24-bit addressing mode can use BASSM to call programs in virtual storage above 16 megabytes.

Uses of BASSM

Programs can use BASSM when branching to modules that might have different addressing modes. In addition, a program called via BASSM can return to its caller in the caller's addressing mode using either BSM or BASSM, provided the called program saves the full contents of the linkage register.

When the second operand is 0 (for example, BASSM 14,0), BASSM:

- Saves the current addressing mode
- Saves the next instruction address
- Does NOT change the PSW AMODE bit
- Does NOT execute a branch

When the first operand is 0 (for example, BASSM 0,14), BASSM does not suppress the saving operation.

Modifying Programs that Invoke Modules Above 16 Megabytes

You must change existing user-written programs that branch to MVS/XA system programs that must be entered in 31-bit addressing mode or via a BASSM instruction. Following are two examples of how you can adapt such programs work in MVS/XA. One method requires that you change user-written programs to use BASSM and BSM instructions. The other method uses a linkage assist routine, which does not require that you change your program.

To understand the following examples, you need to know:

- How BASSM and BSM work. Both are described in "New Branch Instructions."
- That the LOAD macro returns the addressing mode in which the module expects to receive control in the high order bit of register 0.

The modified program in the first example uses LOAD to determine not only the entry point address of the SYSRTN module, but also its addressing mode. The linkage assist routine in the second example uses LOAD to determine the same information for XYZNEW.

Using BASSM and BSM Instructions

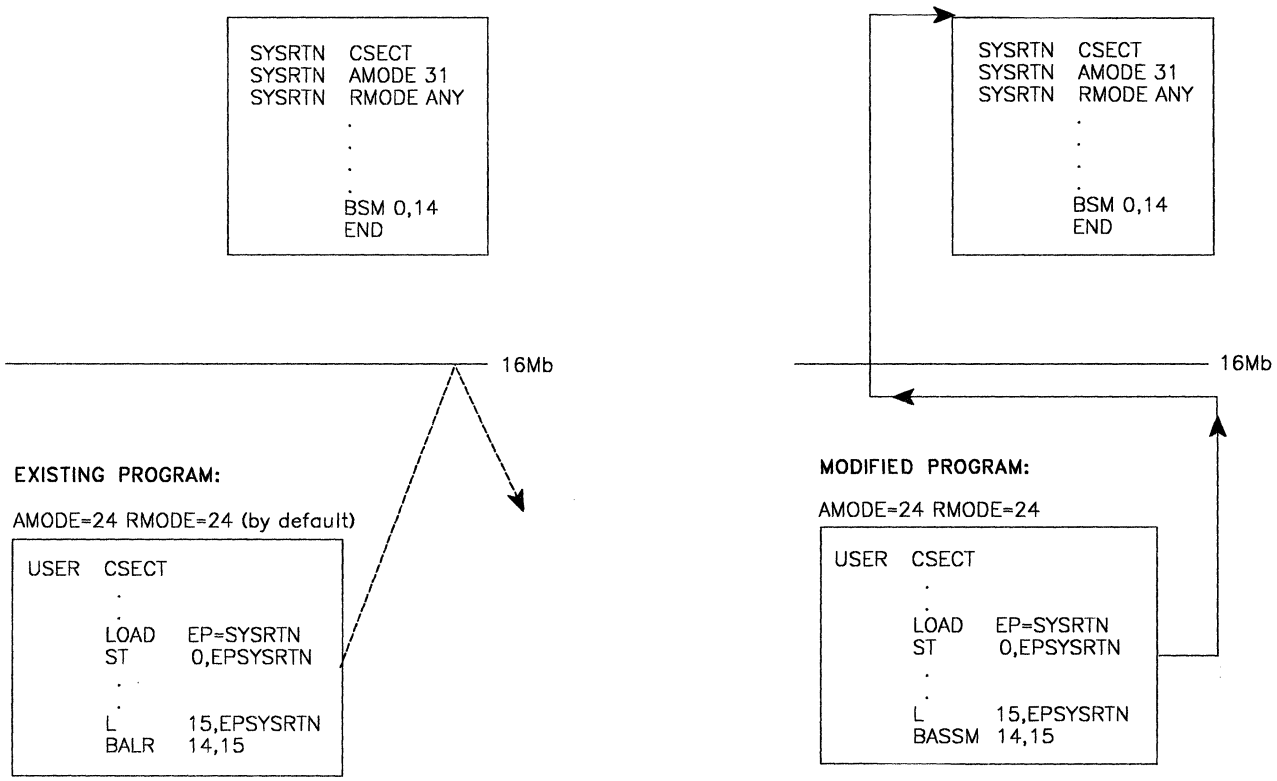
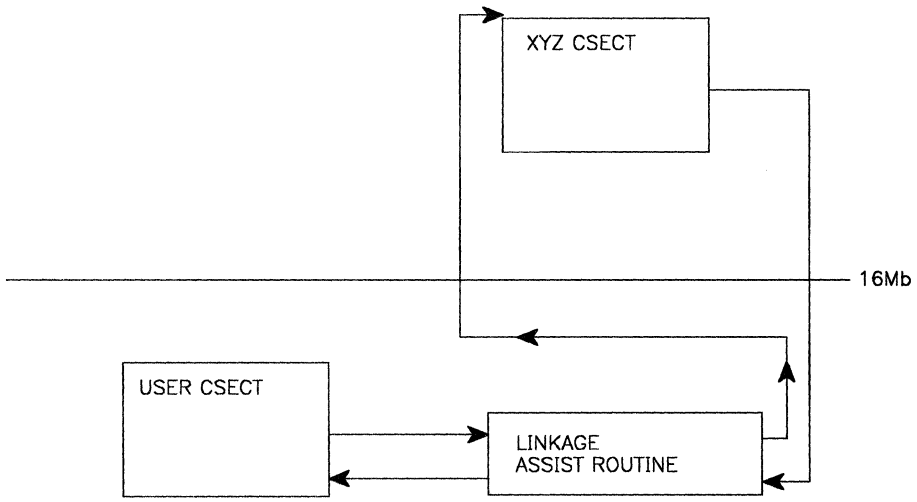


Figure 3-3. Example of Using BSM and BASSM

Linkage-assist routines, also known as glue modules, are programs that handle the mode setting required to pass control between programs in 24- and 31-bit addressing mode:



The linkage assist routine should:

- Save registers.
- Ensure that all parameter addresses and linkage registers contain 31-bit address values.
- Obtain a new save area.
- Branch to the target module using a BASSM instruction.
- Receive control from the target module after it executes.
- Free the new save area.
- Restore registers.
- Return to the caller.

SPL: 31-bit Addressing provides guidelines for using linkage assist routines. Linkage assist routines are mentioned in this publication for two reasons:

- You can use linkage-assist routines to make user-written programs work in MVS/XA without change. Using a linkage-assist routine is practical when:
  - A 24-bit mode program repeatedly calls a program that now executes in 31-bit mode.
  - Several 24-bit mode programs call the same program that now executes in 31-bit mode.

If a user-written program invokes a 31-bit mode program only once, other methods of mode switching might be preferable to using a linkage-assist routine.

- Linkage assist routines invoke system programs that have been moved above 16 megabytes. This change is transparent to most users. Programmers might, however, notice linkage assist routines when tracing control flow during problem determination. Some addresses in the CVT now point to linkage-assist routines instead of to the target module itself.

Following is one example of a linkage-assist routine that passes control between a 24-bit mode user program and a 31-bit mode system program. Using the routine requires renaming the target routine and giving the linkage-assist routine the original name of the target module.

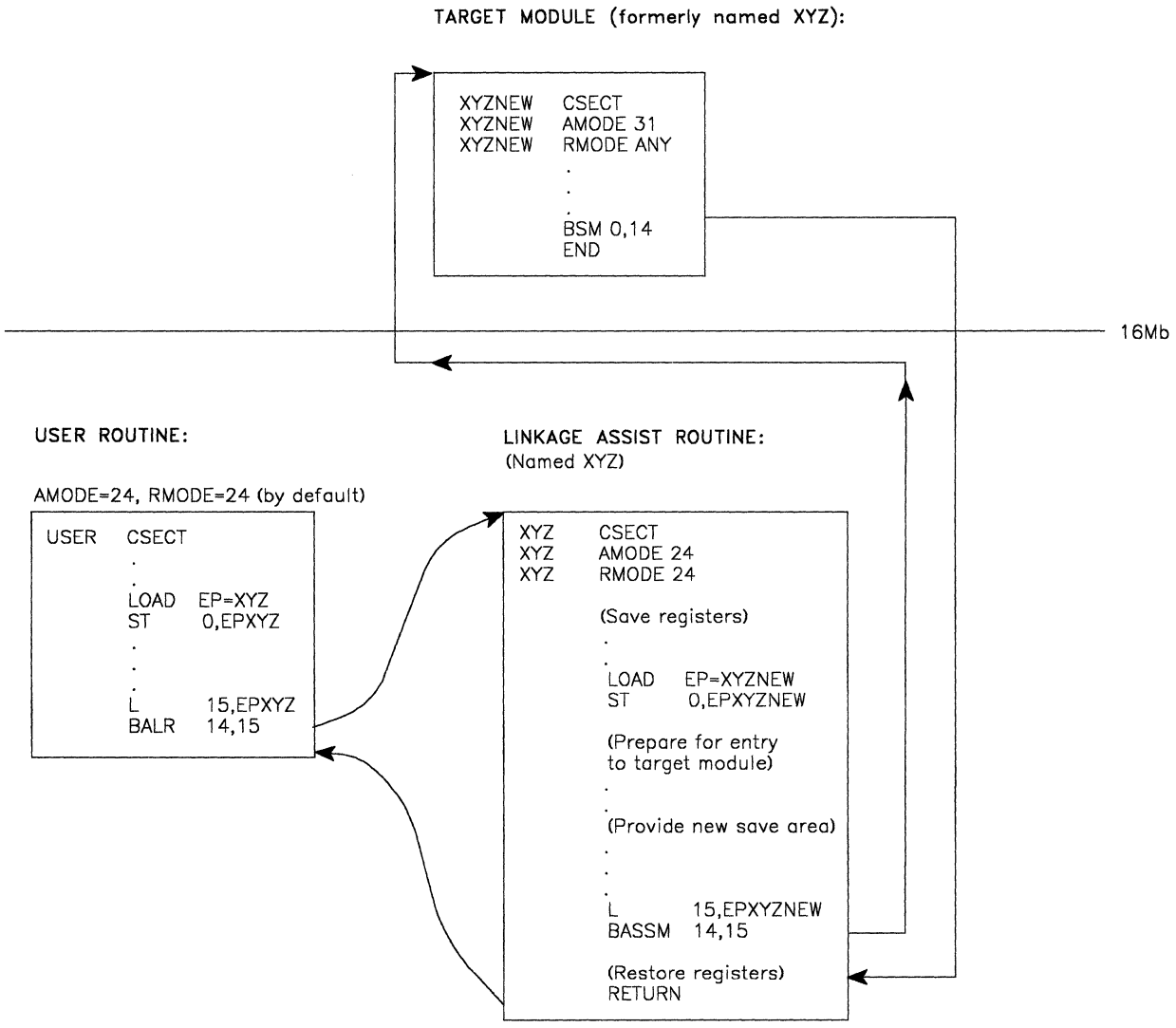


Figure 3-4. Example of a Linkage-Assist Routine

Retrieving Data from a Control Block Above 16 Megabytes

You must change existing user-written programs that access system control blocks that have been moved to virtual storage above 16 megabytes. The following example is one way you can modify those programs. The example requires that you insert mode-setting code before and after the instruction that must be executed in 31-bit addressing mode (L 2,0(,15)).

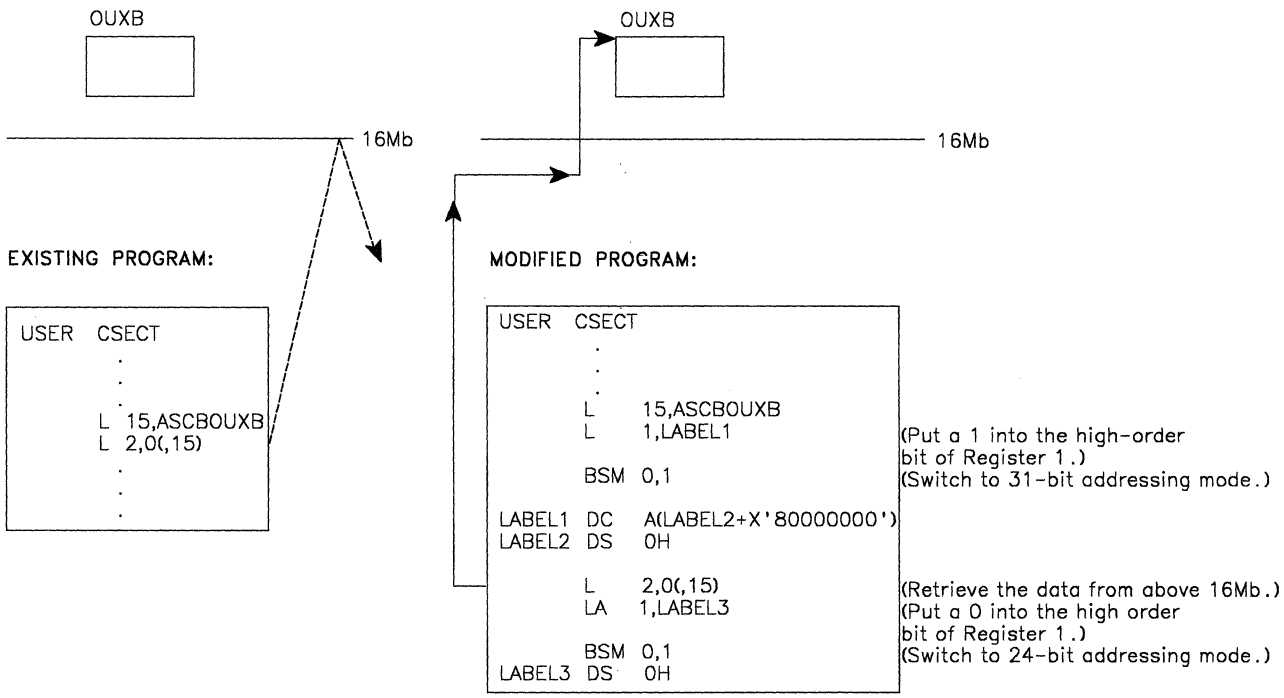


Figure 3-5. Retrieving Data from Above 16 Megabytes

Performing I/O in 31-bit Addressing Mode

To perform I/O, a program executing in 31-bit addressing mode must either:

- Use VSAM services, which accept callers in either 24- or 31-bit addressing mode. (See "Services with Some Restrictions on Address Parameter Values.") Programs using VSAM can access buffers that reside above 16 megabytes.
- Use the EXCP macro. All parameter lists, control blocks, and EXCP appendage routines must reside in virtual storage below 16 megabytes. See "Using the EXCP Macro."
- Use the EXCPVR macro. All parameter lists, control blocks, and appendage routines must reside in virtual storage below 16 megabytes. See "Using the EXCPVR Macro Instruction."
- Use an intermediate routine that executes in 24-bit addressing mode as an interface to non-VSAM access methods, which accept callers executing in 24-bit addressing mode only.

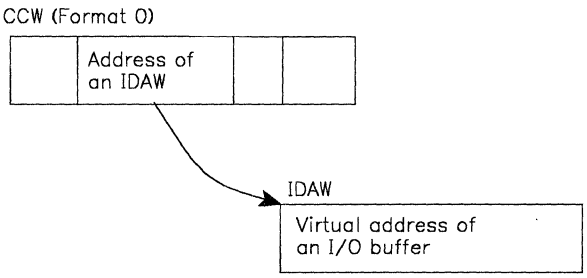
To perform I/O to buffers located in virtual storage above 16 megabytes programs must use either:

- VSAM. Specify on the access method control block (ACB) at OPEN time that I/O buffers are to reside above 16 megabytes.
- The EXCP macro and new virtual IDAW support, which "Using the EXCP Macro" describes.
- The EXCPVR macro. IDAWs can contain real addresses above 16 megabytes, as described in "Using the EXCPVR Macro Instruction."

Using the EXCP Macro

EXCP users can now:

- Back all I/O buffers with real storage above 16 megabytes. To back I/O buffers below 16 megabytes with real storage above 16 megabytes, callers must specify LOC=(BELOW,ANY) on the GETMAIN request. (See "Parameters on the GETMAIN Macro Instruction.")
- Perform I/O to virtual storage areas above 16 megabytes. CCWs in the channel program that EXCP initiates can point to a virtual IDAW. The virtual IDAW contains the 31-bit virtual address of an I/O buffer, which can be anywhere in virtual storage. The EXCP service routine supports only Format 0 CCWs, the CCW format used in MVS/370.



Any CCW that causes data to be transferred can point to a virtual IDAW. Virtual IDAW support is limited to non-VIO data sets. Programmers must be aware of this fact when coding the JCL to execute a program that uses virtual IDAWs.

Although the I/O buffer can be in virtual storage above 16 megabytes, the virtual IDAW that contains the pointer to the I/O buffer and all other areas related to the I/O operation (CCWs, IOBs, DEBs, DCBs, and appendages) must have 24-bit virtual addresses.

See *MVS/Extended Architecture System Programming Library: Data Management* for information on using the EXCP macro.

Summary of New and Updated Macros

Figure 3-6 lists macros that were new or that had new or updated options during Release 2.1.x. Because updates may also have been made to macros in the list during Release 2.2.x, be sure to consult the corresponding chart in the *MVS/XA Conversion Notebook, Volume 2*. When assembling programs that use any of the new function, use the MVS/XA MACLIB. With the following exceptions, the object code generated will be downward incompatible:

- The new LOC, VRC, and VRU options on the GETMAIN macro are downward compatible, as described in "Parameters on the GETMAIN Macro Instruction."

- The MVS/XA MACLIB expansions of SYNCH macros that specify AMODE=24 are downward compatible. However, if the AMODE parameter is omitted or if it specifies any option other than 24, the MVS/XA expansion of SYNCH will not run on an MVS/370 system. See "Downward Incompatible SYNCH Macros" in Chapter 9 for more information.

Macro	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
ABEND	x							A new keyword, REASON, specifies a reason code that supplements the completion code for abnormal termination. The list form of the SNAP macro to which the DUMPOPT parameter points can specify all of the new SNAP dump options.
BLSABDPL			x					A new macro that maps the BLSABDPL exit parameter list. Dump analysis and formatting exit routines that run under IPCS, PRDMP, or SNAP can use BLSABDPL when invoking the new services described in "New Services for Dump Processing Exits" in Chapter 5.
BLSQMDEF and BLSQMFLD			x					A pair of macros that when used together define the structure of a control block. BLSQMDEF marks the beginning of the control block and defines the header. Several BLSQMFLD macros follow, each of which defines a single field in the control block. A second BLSQMDEF macro denotes the end of the control block. The definition is called a control block model. Dump analysis and formatting routines can use the models instead of format patterns for formatting control blocks. See "Format Model Processor Service" in Chapter 5 for more information.
BLSRESSY			x					A new macro that maps one IPCS symbol table record. Dump analysis and formatting routines that run under IPCS and use either the new GET or EQUATE service can use BLSRESSY to describe the record to be retrieved or stored in the symbol table. See "New Services for Dump Processing Exits" in Chapter 5 for more information about the GET and EQUATE services.
CALLDISP					x	x		As long as there is an enabled unlocked task (EUT) FRR in effect, you can use CALLDISP while the LOCAL or CML lock is held. See the discussion of CALLDISP in *System Macros and Facilities*.
CALLRTM	x							A new keyword, REASON, specifies a reason code that supplements the completion code for abnormal termination. The list form of the SNAP macro to which the DUMPOPT parameter points can specify all of the new SNAP dump options.
CHANGKEY	x							Changed to support 31-bit addresses. Addresses given as parameters must be 31-bit addresses regardless of the addressing mode of the CHANGKEY macro user.
CHKPT	x							New keywords: - IDADDR specifies the address of a checkpoint ID - IDLNG specifies the length of the checkpoint ID field - DDNADDR specifies the address of a DDNAME You can use IDADDR and IDLNG instead of the ID and LNG parameters.
CIRB	x							A new parameter, AMODE, specifies the addressing mode in which the specified asynchronous exit routine is to get control. If the AMODE parameter is not specified, the exit routine gets control in the issuer's addressing mode.
CPOOL	x							A new macro that: - Creates a cell pool as described by the requestor - Obtains or returns a cell to a previously constructed cell pool - Deletes a previously constructed cell pool Requestors require authorization only when the storage to be obtained is in an authorized GETMAIN subpool.
CPUTIMER	x							A new macro that obtains the current CPU (processor) timer clock value. CPUTIMER allows users to determine the amount of time remaining in a time interval established by an SRBTIMER or STIMER macro. CPUTIMER uses a PC instruction instead of an SVC. Therefore, it is faster than using TTIMER. Also, users can issue CPUTIMER in task or SRB mode. Issuers of TTIMER must be in task mode.

Figure 3-6 (Part 1 of 5). Summary of New and Updated Macros

Macro	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
DATOFF	x							A new macro that provides linkage to DAT-off routines. See "DAT-off Restrictions."
ENQ		x						Users might receive a new return or abend code. In Release 2.1.1, global resource serialization limits the number of ENQ, RESERVE, and certain GQSCAN requests a single job, started task, or TSO user can have outstanding at a given time. If an address space reaches the limit, the system terminates unconditional ENQ requests with abend code x'538' and rejects conditional requests with a return code of x'014'. See "Limiting Concurrent Global Resource Serialization Requests" for more detail.
			x					ENQ has two new keywords, MASID and MTCB, which are designed for internal use. They are mentioned here only because you might see them in dumps or source code. MASID and MTCB specify the ASID and TCB address of a task, respectively. They are useful in situations where one task is performing work for another and might require resources that task owns. If an ENQ specifying MASID and MTCB fails because another task owns the resource, the task issuing the ENQ can determine whether the identified task is the owner. New return codes provide that information. If the specified task owns the resource, the issuing task can choose to use the resource anyway. Both tasks, however, must have previously established an alternate method of serialization.
ESPIE	x							A new macro that provides services similar to the SPIE macro for callers in either 24- or 31-bit addressing mode. SPIE users must be in 24-bit addressing mode.
EXCP	x							Changed to support the use of real and virtual storage above 16 Mb. See the section, "Using the EXCP Macro," in this Chapter.
EXCPVR	x							Changed to support 31-bit addressing and the use of real storage above 16Mb. See "Using the EXCPVR Macro Instruction" in this Chapter.
GETMAIN	x							Three new parameters: VRC, VRU, and LOC. See "Parameters on the GETMAIN Macro Instruction."
GQSCAN		x						Users might receive a new return code. In Release 2.1.1, global resource serialization limits the number of ENQ, RESERVE, and certain GQSCAN requests a single job, started task, or TSO user can have outstanding at a given time. If an address space reaches the limit, global resource serialization rejects subsequent GQSCAN requests that specify the TOKEN option and request more information than can fit into the caller's buffers. The issuer receives a return code of x'14.' Global resource serialization returns the buffers of information, but does not continue the scan. For more information, see "Limiting Concurrent Global Resource Serialization Requests."
			x					Beginning with Release 2.1.2 you can specify generic (partial) rnames and qnames on the RESNAME keyword. GQSCAN attempts to obtain information about any resource that matches the specified part. The keywords and parameters that provide the new support are GENERIC, SPECIFIC, rname length, and qname length. GENERIC indicates that the qname and rname on GQSCAN are generic names. The qname and rname lengths specify the number of characters in the qname or rname that must match the qname and rname specified on GQSCAN. SPECIFIC indicates that the complete qname and rname must match. It is the default.
GTRACE	x							A new parameter, TEST, determines whether data gathering and tracing are to be done. See "Using GTF to Trace User Events." *SPL: Service Aids* describes GTRACE.

Figure 3-6 (Part 2 of 5). Summary of New and Updated Macros

Macro	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
IOSINFO				x				A new macro that obtains the subchannel number for a specified unit control block (UCB). The macro returns the subsystem identification word (SID) which identifies the subchannel number of the UCB in a user-specified location.
IOSLOOK	x							A new macro that replaces other methods of locating UCBs. See "IOSGEN UCBLOOK Macro Instruction" in this Chapter.
LOAD	x							New keywords: - EOM=NO|YES specifies whether a module in global storage is to be deleted at address space termination (YES) or end-of-task time (NO). The default is NO. Previously, the system deleted the modules at end-of-task time. - LOADPT requests that the virtual address of the first byte of the load module be returned at the location indicated on the LOADPT keyword. The high-order bit of the entry point address that LOAD returns indicates the addressing mode in which the routine expects to receive control.
MGCR			x					Beginning with Release 2.1.2, you can use MGCR to issue internal REPLY commands, as well as internal START commands. This new function enables you to write WTO/WTOR exits that respond to particular WTOR macros. *SPL: User Exits* shows an example of such an exit. (WTO/WTOR exits are also new in Release 2.1.2. See "New WTO/WTOR User Exits" for more information.)
NUCLKUP	x							A new macro that retrieves either: (a) the address and addressing mode of a CSECT or entry point in the DAT-on nucleus, or (b) the name and address of a CSECT that resides at a given address in the DAT-on nucleus.
PGSER	x							A new macro that performs the same services as PGANY, PGFIX, PGFIXA, PGFREE, PGFREEA, PGLOAD, PGOUT, and PGRLSE. PGSER can use 31-bit addresses, the other services listed cannot.
PTRACE	x							A new macro that creates system trace table entries.
RACROUTE	x							A new macro that requests the RACF services that FRACHECK, RACDEF, RACLIST, RACHECK, and RACINIT invoke. The REQUEST parameter on RACROUTE indicates which service is to be performed. Any of the parameters that are valid on the other RACF macros are also valid on RACROUTE. Thus, using RACROUTE is very similar to using the other macros. Note one difference, however. RACROUTE requires a 512-byte work area, while the other RACF macros do not.
RESERVE		x						Users might receive a new return or abend code. In Release 2.1.1, global resource serialization limits the number of ENQ, RESERVE, and certain GQSCAN requests a single job, started task, or TSO user can have outstanding at a given time. If an address space reaches the limit, the system terminates unconditional RESERVE requests with abend code x'538', and rejects conditional requests with a return code of x'014'. See "Limiting Concurrent Global Resource Serialization Requests" for more detail.
			x					Release 2.1.2 adds two new keywords, MASID and MTCB. See the ENQ entry for more information.
RETURN		x						Release 2.1.1 changes the flag that the T parameter requests the system to set. The new flag is the low-order bit of the fourth word in the called program's save area. The system sets that bit to one after the called program has returned to its caller. In previous releases, the same flag is the high order byte of the fourth word in the save area.

Figure 3-6 (Part 3 of 5). Summary of New and Updated Macros

Macro	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
SDUMP	x							New parameters: - ALLNUC, a new option on the SDATA keyword, requests that the DAT-off nucleus and the entire DAT-on nucleus be dumped. - SUBPLST and KEYLIST specify the subpool and key of storage to be dumped. - TYPE=NOLOCAL indicates that SDUMP is not to obtain a local lock. Changed parameters: - NUC requests that the DAT-on, non-page-protected section of the nucleus be dumped. In MVS/370, NUC causes the entire nucleus to be dumped. The ALLNUC option requests the entire MVS/XA nucleus. - TRT requests trace data from the active trace facilities, as in MVS/370. However, in MVS/XA, if the caller is unauthorized, the dump includes system trace data for the caller's address space only. MVS/XA dumps for authorized requestors and all MVS/370 dumps include the entire system trace table. Also, unlike MVS/370, MVS/XA dumps can include both system trace and GTF data, because both trace facilities can be active at the same time. If SDUMP specifies any new parameters, the macro must be assembled using the MVS/XA macro expansion. If it is not, the system ignores the new parameters and flags the others as errors.
		x						As of Release 2.1.1 SDUMP dumps 4 K of storage before and 4 K after the addresses in the PSWs and registers stored in the IHSA, SDWA, and PSA. Earlier releases dump only 2 K of storage before and 2 K after those addresses.
SETLOCK	x							New parameters support the new lock types. Programs that issue SETLOCK RELEASE,TYPE=(reg)|ALL must use the MVS/XA expansion of SETLOCK in some situations. See "SETLOCK RELEASE,TYPE=(reg)|ALL Macro Instruction."
SETRP	x							A new keyword, REASON, specifies a reason code that supplements the completion code for abnormal termination. The list form of the SNAP macro to which DUMPOPT points can specify all of the new SNAP dump options.
SMFIOCNT	x							A new macro that supplies to SMF either the EXCP count, the device connect time, or both.
SNAP	x							New parameters: - SUM, a new option on the SDATA keyword, requests a new summary dump. See "User Summary Dumps" in Chapter 6. - SUBPLST requests individual subpools. When SNAP specifies the SUBPLST option, the length of the list and standard forms of the macro expansion increase. The object code generated is downward incompatible. - ALLVNUC, a new option on the SDATA keyword, requests that the entire DAT-on nucleus be dumped. In SYSMDUMPs, ALLVNUC causes the entire nucleus to be dumped. - SUBTASKS, a new option on the PDATA keyword, requests that program data for all subtasks of a designated task be dumped. Changed parameters: - NUC requests that the DAT-on, non-page-protected section of the nucleus be dumped. SYSABEND dumps also include the PSA and CVT. In MVS/370, NUC causes the entire nucleus to be dumped. - TRT requests trace data from the active trace facilities, as in MVS/370. In MVS/XA, the dump can include system trace data and GTF data. In MVS/370, system trace and GTF cannot be active at the same time. Therefore, MVS/370 dumps never include both system trace and GTF data. - SDATA=ALL requests all of the SDATA options except ALLVNUC. In MVS/370, it includes all SDATA options.

Figure 3-6 (Part 4 of 5). Summary of New and Updated Macros

Macro	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
SPLEVEL	x							A new macro that controls which expansion of a macro the assembler generates. The MVS/XA MACLIB contains both MVS/XA and MVS/370 expansions of macros whose MVS/XA expansions do not work in MVS/370. See "Handling Downward Incompatible Macros" in Chapter 9.
STIMERM			x					A new macro that sets a timer for a real time interval, as does the existing STIMER macro. STIMERM is different from STIMER, however, in that: - Each task can have up to 16 STIMERM intervals in effect at the same time. Only one STIMER interval is allowed. - STIMERM sets only real time intervals; STIMER sets both task and real time intervals. - STIMERM can also test how much time remains in the interval and cancel the interval. TTIMER provides those functions for STIMER intervals. - STIMERM can pass a 4-byte parameter to the exit routine that receives control when the interval expires. STIMER cannot. STIMERM is provided so that a task can easily have a task interval and one or more real time intervals in effect at the same time. A task can, for example, set an STIMER interval to measure task time and an STIMERM interval to simultaneously measure real time.
SVCUPDTE	x							A new macro that allows authorized programs to dynamically update the SVC table. In MVS/370, defining an SVC after SYSGEN requires an IPL of the system.
SYNCH	x							A new parameter, AMODE, indicates the addressing mode in which the specified program is to get control. If the AMODE parameter is not specified, the program gets control in the issuer's addressing mode. Unless SYNCH specifies the AMODE=24 parameter, programs that use SYNCH and are assembled using the MVS/XA MACLIB will not run on MVS/370 systems. See "Downward Incompatible SYNCH Macros" in Chapter 9 for more information.
VSMLIST	x							A new macro that returns information about virtual storage allocation within an address space.
VSMLOC	x							A new macro that verifies a given virtual storage area has been allocated to satisfy a GETMAIN request.
VSMREGN	x							A new macro that returns the starting addresses and sizes of the private area regions associated with a given TCB in the current address space.
WTL			x					A new keyword, OPTION=PREFIX or OPTION=NOPREFIX, indicates whether the WTL text contains a prefix to identify the log record. If the text already contains a prefix, specify the PREFIX option. If you specify NOPREFIX or omit the OPTION keyword altogether, the system inserts a two-character prefix. ('X' is the default prefix.)
WTO		x						You can specify routing code 11 (programmer information) on multiline WTO messages, an option earlier releases do not allow. If, because of that restriction, you have programs that issue a series of single line WTOs with the same message ID, you can improve performance by combining the messages into one multiline message. You can also use the HRDCOPY option when writing multiline WTOs. In earlier releases you cannot write multiline messages to hardcopy only.
			x					A new CMD option on the MCSFLAG keyword enables you to record system commands in the system log. MCSFLAG=CMD indicates that the text is a system command and requests that it be entered in the system log.
WTOR			x					Release 2.1.2 adds the CMD option on the MCSFLAG keyword. See the WTO entry.

Figure 3-6 (Part 5 of 5). Summary of New and Updated Macros

Parameters on the GETMAIN Macro Instruction

MVS/XA provides three parameters on GETMAIN: VRC, VRU, and LOC.

VRC and VRU Parameters

VRC (variable request conditional) and VRU (variable request unconditional) are two new forms of GETMAIN. Both issue SVC 120.

```
GETMAIN VRC,LV=(maximum length,minimum length)
GETMAIN VRU,LV=(maximum length,minimum length)
```

VRC and VRU request a single area of virtual storage having a length between the maximum and minimum lengths specified. MVS/XA returns the address of the allocated virtual storage in Register 1 and the length of the storage in Register 0.

Callers in 24- or 31-bit addressing mode can use VRC or VRU. However, MVS/XA treats all parameter lengths and addresses as 31-bit values.

VRU and VRC are exceptions to the general rule that programs using new MVS/XA function are not downward compatible. Both generate object code that runs on MVS/370 systems. MVS/370 treats VRC and VRU parameters as RC and RU, respectively, and obtains the maximum length of storage specified on the LV operand. MVS/370, of course, also uses 24-bit parameter values.

LOC Parameter

The LOC parameter has two subparameters for specifying whether virtual storage is to be obtained above or below 16 megabytes and how it is to be backed if fixed (below 16 megabytes or anywhere). (RSM always allocates real storage anywhere until the storage is fixed.) Possible LOC specifications are:

LOC=(BELOW)	VSM must allocate virtual storage below 16 megabytes.
LOC=(ANY)	VSM can allocate virtual storage anywhere.
LOC=(RES)	VSM allocates storage according to the requestor's residence. If the requestor resides in virtual storage below 16 megabytes, VSM allocates storage below 16 megabytes. If the requestor resides above 16 megabytes, VSM allocates storage anywhere.
LOC=(parm1,ANY)	RSM attempts to back the page with real storage above 16 megabytes. If unsuccessful, RSM backs the page with real storage below 16 megabyrtes. Note that, regardless of the LOC specification, RSM backs virtual storage with real storage anywhere until the storage is fixed (either by definition or by a PGFIX or PGSER macro). The first subparameter (parm1) can be BELOW, ANY, or RES.

LOC is especially useful in programs with 24-bit dependencies. Programs that reside above 16 megabytes must specify LOC=(BELOW) on requests for storage that has 24-bit dependencies.

You can specify LOC only on the RU, RC, VRU, and VRC forms of GETMAIN (SVC 120). VSM satisfies all other forms with virtual storage below 16 megabytes, which RSM backs with real storage below 16 megabytes.

Like VRU and VRC, LOC is downward compatible. Regardless of the LOC specification, however, MVS/370 always obtains storage below 16 megabytes.

SDUMP Macro Instruction

If you use any new SDUMP keywords, be aware that SDUMP generates a longer parameter list. In addition, once the longer list is assembled in a module, the assembler generates the long form of all subsequent SDUMP parameter lists in the module, regardless of which keywords the SDUMP macros specify. The long list is similar to the short list, except that it has additional bytes appended to the end.

If you do not want the long form to be generated on subsequent macros, use the SPLEVEL macro to request the MVS/370 expansion. See "Handling Downward Incompatible Macros" in Chapter 9.

SETLOCK RELEASE,TYPE=(reg)|ALL Macro Instruction

If any new locks are held when the SETLOCK RELEASE,TYPE=ALL macro is issued, you must use the MVS/XA expansion. The MVS/370 expansion does not recognize the new locks and, therefore, does not release them. Likewise, if the SETLOCK RELEASE,TYPE=(reg) macro is issued and the bit string in the register specifies a new lock, you must use the MVS/XA expansion. If you use the MVS/370 expansion, the system does not release the new locks. You can use an SPLEVEL macro to request either expansion. See "Handling Downward Incompatible Macros" in Chapter 9.

Using GTF to Trace User Events

MVS/XA provides an alternate way of using GTF to trace USR events. Using the new method:

- Applications no longer need to supply and support an external interface that requests tracing. Starting GTF with the appropriate USRP options specified is sufficient to allow applications to trace their own USR events. (The USRP option is new in MVS/XA. See the TRACE entry in Figure 4-3 in Chapter 4.)

  In MVS/370, applications have to provide their own interface for requesting USR event tracing. One example of an interface is the DIAGNS=TRACE subparameter of the DCB parameter on a DD statement, which requests module flow tracing through OPEN, CLOSE, and EOV. Also, any program can support and request tracing of their own USR events by specifying a trace keyword in the PARM field of the EXEC statement.

- Applications can use the new TEST keyword on the GTRACE macro to determine whether or not GTF tracing is active for their USR events. Depending on the return code from GTRACE, applications can either gather the trace data and have it written or bypass tracing.

In MVS/370, applications have to make two tests before issuing GTRACE, one to determine if the application has requested tracing, and another to determine if GTF is active for USR tracing.

The MVS/370 method of establishing tracing capability continues to work in MVS/XA. GTRACE TEST offers an alternate method for MVS/XA users.

The way that applications build the data records to be traced and issue the GTRACE macro to write them to the SYS1.TRACE data set has not changed.

See *SPL: Service Aids* for more information on using GTRACE.

Unit Verification

Three modules in MVS/XA provide unit verification:

1. IEFAB4UV for authorized (key 0 - 7) 24-bit callers
2. IEFGB4UV for authorized (key 0 - 7) 31-bit callers
3. IEFEB4UV for all unauthorized (key 8-15 and task mode) callers

The MVS/370 device allocation tables (DEVNAMET, IEFDEVPT, and DEVMASKT) and module IEFSCAN have been deleted. You must change programs that call IEFSCAN or that directly access the device allocation tables to use any of the three MVS/XA modules that provide unit verification.

IEFAB4UV and IEFGB4UV

Beginning with Release 2.1.1, module IEFGB4UV contains the unit verification function and resides above the 16 megabyte address. IEFAB4UV is a linkage-assist routine that enables programs executing in 24-bit mode to access IEFGB4UV.

The MVS/XA version of IEFGB4UV provides all of the services that the MVS/370 version of IEFAB4UV provides, all of the IEFSCAN services, and some new functions. Specifically, IEFGB4UV can:

1. Check whether the device numbers supplied as input are all associated with the same group.

2. Check whether the device numbers supplied as input are associated with the unit name specified in the eligible device table (EDT).

3. Return the unit name associated with an input value such as a device type. The unit name is the EBCDIC representation of the IBM generic device name (for example, 2305) or the user-defined esoteric name (for example, TAPE).

4. Return the UCB addresses associated with a specified unit name.

5. Return group identification for each input UCB.

6. Indicate whether a specified unit name is an internal representation of the unit name (that is, whether the unit name is an index into the EDT). This service is used with 2 and 4.

7. Return the internal representation of a specified unit name, which can then be used as an index into the EDT.

8. Convert a four-byte UCB device type to an internal representation of a unit name, which can then be used as an index into the EDT.

9. Return general information about a specified unit name, including:

   - Whether the unit name is esoteric, VIO eligible, contains 3330V units, or contains teleprocessing class devices
   - The number of device classes in the unit name
   - The number of generic device types in the unit name

10. Indicate that the parameter list should not be altered, thereby allowing the parameter list to be in storage that is not protected by key 1. This service is used with 2.

IEFEB4UV

IEFEB4UV performs the functions described in 3, 4, 6, 7, 8, and 9 of the IEFGB4UV description for programs in user key and task mode.

See *SPL: System Modifications* for information on using IEFAB4UV or IEFEB4UV.

Programs Using the Vector Facility Enhancement (VFE)

Programs compiled by the Vector FORTRAN compiler or assembled by Assembler H Version 2.1 can make use of the IBM 3090 Vector Facility. The following restrictions apply to programs issuing a vector instruction:

- They must be running in task mode.
- They must run with the processor unlocked and enabled for I/O and external interrupts.

Even if these restrictions are satisfied, the program issuing vector instructions may abnormally terminate if:

- No processor with the Vector Facility has ever been online during an IPL.
- None of the online processors with the Vector Facility installed meets the processor affinity requirements of the task specified in the program properties table (PPT). For this reason, installations with processor complexes that include the Vector Facility should not specify processor affinity for programs in the PPT.
- The required resource environment cannot be created. (For example, it is not possible to issue a GETMAIN macro and acquire enough storage for the necessary vector save areas.)

IMS Applications and the Extended Recovery Facility (XRF)

When you install XRF, there is no need to rewrite IMS user-application programs if they conform to IMS standards. You must be sure, however, that all required data is duplicated on the active and alternate systems and that the required data bases and log files reside on DASD shared by both systems. *XRF Planning* has a detailed description of what to do when installing XRF.

Chapter 4. Operating Considerations

This chapter contains information that pertains to operators and operational procedures. System programmers might also be interested in some of the command changes it describes (for example, SLIP, DISPLAY DUMP, and DUMP). The topics included in this chapter are:

Note: Throughout this chapter you will see references to the "*Operator's Guide* for your type of processor complex." The full title of the *Operator's Guide* depends upon the type of processor you have. The complete titles are listed below:

- For the 308x processor complexes:
  - *IBM 3081 Operator's Guide for the System Console* (GC38-0034)
  - *IBM 3083 Operator's Guide for the System Console* (GC38-0036)
  - *IBM 3084 Operator's Guide for the System Console* (GC38-0037)
- For the IBM 3090 processor complex:
  - *IBM 3090 Processor Complex Operator Controls for the System Console* (SC38-0040)
  - *IBM 3090 Model 200 Processor Complex Operator Tasks for the System Console* (SC38-0041)
  - *IBM 3090 Model 150 and 180 Processor Complex Operator Tasks for the System Console* (SC38-0049)
  - *IBM 3090 Model 400 Processor Complex Operator Tasks for the System Console* (SC38-0050)

- For the 4381 processor complex:
  - *IBM 4381 Processor Operations Manual* (GA24-3949)

Processor Complexes with the Extended Recovery Facility (XRF)

Operator awareness is the key to successful management of a processor complex using XRF. These installations function with an **active** environment (including a processor, Release 2.1.3AE or Release 2.1.7, VTAM, and IMS) that processes IMS transactions. If the active fails, a duplicate, **alternate**, environment takes over. During a takeover, MVS/XA (through the availability manager component), and IMS send messages to the system operator concerning the progress of the takeover. The MVS/XA, IMS, and network operators perform some tasks and must be ready to communicate with one another.

Aside from planned takeovers initiated by the operator, XRF takeovers occur in response to IMS abends, or failures in MVS/XA, XRF processors, VTAM, or the IMS resource lock manager (IRLM). However, other events might also cause XRF takeovers and require operator actions. An XRF takeover might occur:

1. When the operator stops or suspends a processor

   Any actions that stop or suspend a processor can cause an XRF takeover. If the takeover is not desired, the IMS master terminal operator (MTO) should first suspend surveillance.

   Operator actions that stop a processor include:

   - Issuing a SWITCH SYSTEM command
   - Issuing a STOP to a processor
   - Taking an SVC dump of the address space of the active IMS subsystem
   - Setting a SLIP trap that causes suspension of the system

2. When the system stops or suspends a processor

   System-initiated suspensions can cause an XRF takeover even if the resulting system-initiated recovery is successful. The operator does not control these occurrences but will need to take some action such as re-starting the former active as the new alternate IMS system.

   Examples of system-initiated suspensions are:

   - Restartable WAITs (or disabled console interactions)
   - System-initiated SVC dump of the IMS subsystem address space
   - Processor or operating system recovery actions that take longer than the installation-specified threshold

XRF Planning gives an overview of installing and operating the XRF system. This book also provides references to books with specific instructions for using the components of XRF. The components include:

- Release 2.1.3AE or Release 2.1.7
- IMS/VS Version 2
- ACF/VTAM Version 3
- MVS/XA DFP Version 2
- ACF/NCP Version 4

Jobs Waiting for the Vector Facility

When jobs issue a vector instruction and a processor with the Vector Facility is not available, the system swaps them out and sends an informational message (IRA700I) to the operator console. You can find out if a job is in such a "vector wait" by giving the DISPLAY system activity, ALL command (DISPLAY A, A). The AFF field contains *VF* if the job is waiting for a processor with the Vector Facility.

Loading 370-XA Microcode

Before you can run MVS/XA you must load the 370-XA microcode. The procedure for doing this depends upon which processor complex you have. Refer to the *Operator's Guide* for your type of processor complex to find appropriate 370-XA load procedures.

SYSCTL (SCP Manual CNTL) Console Frame

A SYSCTL console frame comes with 370-XA processors. The SYSCTL frame is very similar to the SC frame available on 3036 consoles attached to 3033 processors. The frame allows the operator to request some of the functions offered on the existing OPRCTL (operator control) frame, as well as some additional functions.

Depending upon the type of 370-XA processor you have, your processor supports some or all of the SYSCTL console frame functions described below. Consult the *Operator's Guide* for your type of processor complex to find out exactly which ones are supported on your processor complex. The SYSCTL frame allows the operator to:

- Specify an alternate nucleus. Operators should use the SYSCTL frame instead of previous methods for specifying an alternate nucleus. Previous methods include:
  - Using the OPRCTL frame or the SC frame.
  - Storing an alternate nucleus identifier in absolute location 8 after the hardware IPL completes and while the processor is in instruction step mode. VM users must still use this method.
  - Specifying 'ALT=xx' when asked to specify system parameters at IPL time. MVS/XA does not support the ALT parameter.

- Specify the device number from which to IPL the operating system.

- Specify the device number from which to IPL the stand-alone dump program. Having the stand-alone dump IPL option separate from the operating system IPL option prevents the operator from inadvertently loading the wrong program. In addition, issuing the stand-alone dump IPL command from an SYSCTL frame causes the hardware to automatically store status if the auto-store-status indicator is on.

- Specify how MVS/XA is to perform restart processing. The operator can request that MVS/XA:

  Option 0 - Display information about the work in progress. The operator can then choose to either terminate the interrupted work and invoke the necessary recovery routines, or return to the point of interruption.

  Option 1 - Attempt to diagnose and repair the problem. In response, the system takes several corrective actions.

  With an OPRCTL frame, the operator does not have an option. MVS performs restart processing as described in option 0 for the SYSCTL frame.

  When performing restart processing, operators should use the SYSCTL frame. If the system is in a restartable wait state, operators should either:

  - Select option 0 on the SYSCTL frame. Operators must not select any other option.

  - Use the restart button. Using the restart button is a valid option only if the operator knows which processor is the target of the restart. The current frame might not identify the target processor, in which case the hardware uses a previously-established target.

  - Use a bottom line command. The bottom line command allows the user to specify a target processor and can be used with any frame.

  - Use the OPRCTL frame.

- Request instruction recording. The operator can have the hardware record instruction addresses on a disk in the service processor. The operator can obtain a hard copy using stand-alone dump or SVC dump formatted by print dump. Previously, when a program was looping, the operator had to record by hand the instruction counter addresses before taking a dump.

- Allow instruction stepping on both processors.

Storing Status Before Taking a Stand-Alone Dump

For the first IPL of stand-alone dump after an MVS/XA IPL, if the operator initiates the IPL from a hardware display frame, the hardware automatically stores status. The operator is not required to store status manually.

Using Labeled Tapes for Stand-Alone Dumps

Stand-alone dump (SADMP) can use labeled tapes that are not password-protected. If the operator mounts such a tape, SADMP displays the VOLSER and asks the operator if the tape is to be used. Note, however, that **SADMP writes over the label**. If used again as a labeled tape, the tape has to be re-labeled.

In MVS/370, SADMP rejects all labeled tapes.

JCL Changes to Jobs that Allocate SYS1.DUMP Data Sets

Jobs that allocate SYS1.DUMPnn data sets (for example, AMDPRDMP or IEBGENER to unload dump data sets) must specify DISP=SHR on the JCL. You must change DISP=OLD to DISP=SHR on any DD statements that allocate SYS1.DUMPnn data sets.

MVS/XA now allocates dump data sets to the DUMPSRV address space to improve integrity. The data sets are allocated with DISP=SHR and DUMPSRV does not free them after taking a dump. Thus, jobs that request SYS1.DUMPnn data sets with DISP=OLD cannot access them.

Processing Hot I/O Interrupts

Hot I/O interrupt processing is changed. (Hot I/O interrupts are consecutive, unsolicited interrupts on a subchannel. They are caused by hardware malfunctions.) In MVS/XA:

- IOS uses a single criterion for detecting hot I/O conditions. Because of the new I/O architecture, no other thresholds are necessary. In MVS/370, IOS uses separate thresholds to detect excessive time-outs and hot I/O conditions on channels, devices, and control units.

- Installations can specify hot I/O recovery actions that IOS performs automatically. Unlike MVS/370, recovery does not have to involve the operator.

You can specify recovery actions for the following device categories:

- Reserved DASD
- Non-reserved DASD
- All other devices

Valid recovery actions are:

- Asking the operator to direct recovery (as in MVS/370).

- Boxing the hot device (forcing the device offline in such a way that future I/O requests for the device are returned to the I/O driver as permanent errors).

- Performing channel path recovery.

- Forcing the channel path offline.

The following figure shows the IBM-supplied recovery actions:

Device Category	Non-recursive Condition	Recursive Condition
RESERVED DASD	Request direction from the operator	Request direction from the operator
NON-RESERVED DASD	Perform channel path recovery	Request direction from the operator
ALL OTHER	Request direction from the operator	Request direction from the operator

Figure 4-1. Default Hot I/O Recovery Actions

The hot I/O threshold and recovery actions are contained in a new module, IOSRHIDT, which replaces IECVHIDT. If Release 2.1.1 or a later release is installed, you can change the defaults using the new HOTIO keyword in the IECIOSxx PARMLIB member. If the system is at the Release 2.1.0 level, use the AMASPZAP service aid instead. *SPL: System Modifications* describes how to change the defaults in more detail.

Extended Color Support on 3279 Multiple Console Support (MCS) Consoles

MVS/XA provides additional ways of highlighting messages on 3279 MCS color consoles, Models 2B and 3B. You can:

- Display message types or console fields in up to seven different colors. (MVS/SP Version 1 Release 3 provides four-color support for 3279 MCS color consoles. Four-color support is still available on Models 2A, 2C, and 3A.)

- Highlight messages with underscoring, blinking, or reverse video display (black characters on a colored background).

Color-coding and other highlighting techniques help operators distinguish the importance of messages. As such, highlighting can be an effective way of controlling message traffic.

You can specify highlighting attributes for your installation in an MPFLSTxx PARMLIB member. If none are specified, the system uses default values. The same highlighting attributes are in effect for all 3279 consoles, Models 2B and 3B. A SET MPF=xx command, where xx identifies an MPFLSTxx PARMLIB member, causes the system to use the attributes in the specified MPFLSTxx member. When highlighting attributes are changed, the system puts the name of the old MPFLSTxx member in the SET MPF command section of a type 90 SMF record. The DISPLAY MPF command displays the current specifications. *SPL: Initialization and Tuning* describes how to use MPFLSTxx.

Controlling Message Traffic on Operator Consoles

In general, as the system workload increases, messages appear on the operator console at a faster rate. To keep the message rate manageable, your installation needs to evaluate its current methods of tailoring message output. Methods of controlling message traffic include using:

- The WTO/WTOR user exits, which are new in Release 2.1.2, or the existing WTO user exit (IEECVXIT). The new WTO/WTOR exits can modify processing for any message. They can also alter processing in more ways than IEECVXIT can. See "New WTO/WTOR User Exits" in Chapter 5 for more information.

- Additional operator consoles with multiple console support (MCS). *System Commands* describes how to use MCS consoles.

- The message processing facility, which suppresses nonessential messages from the operator console. *SPL: Initialization and Tuning* describes how to use the message processing facility.

- Console clusters, which reduce message traffic on a single console. *System Commands* describes how to use console clusters.

- The TRACK command to display system status, instead of having JOB STARTED/ENDED messages displayed on the console. *System Commands* describes the TRACK command.

- Message routing codes to direct application messages to the appropriate console. *System Commands* describes how to assign routing codes.

- Color displays to help operators distinguish important messages. Four-color message coding is available on 3279 consoles, Models 2A, 2C, and 3A; seven-color message coding is available on 3279 consoles, Models 2B and 3B. *System Commands* and *SPL: Initialization and Tuning* describe how to assign color attributes.

- Precise inquiries. By requesting only pertinent data, operators can reduce message volume. For example, by using D U,,,130,1 instead of D U to display the status of device 130, the operator generates 3 lines of output instead of 52 (a default maximum). *System Commands* documents these techniques.

- A console other than the master console to start tasks, such as VTAM, that communicate with the console on which they are started. VTAM retains the ID of the console from which the START VTAM command is issued and directs all messages to that console. Therefore, if possible, start VTAM on a console defined to receive TP messages to reduce traffic on the master console.

- A terminal dedicated to RMF. Use RMF Monitor II reports to determine what the system is doing. See *RMF Reference and User's Guide* for more information.

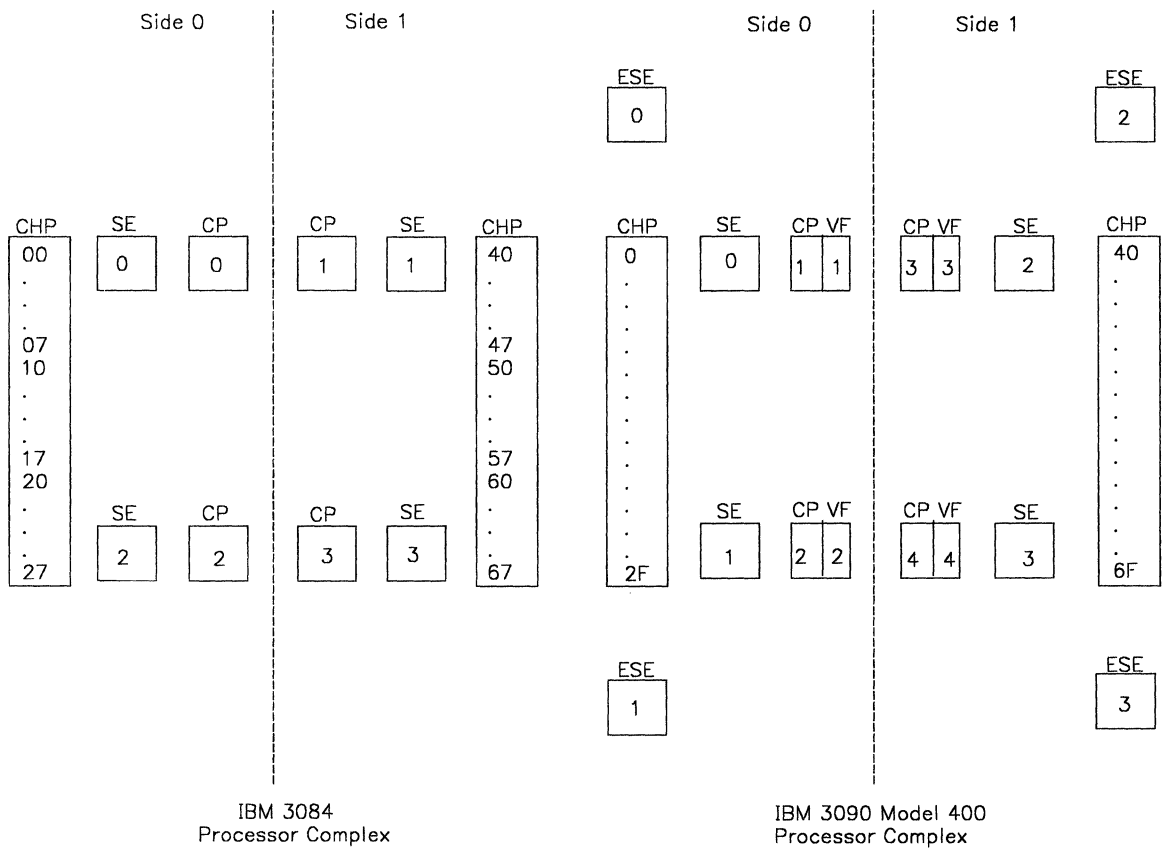
Response to Message IOS201E

MVS/XA issues message IOS201E after it recovers from an error condition that required it to stop processors that shared a resource. The message indicates whether or not the resource was lost (that is, whether a task was in the process of updating the resource and lost its reserve before the update was finished). After Release 2.1.1 is installed, operators must reply U to message IOS201E before the system restarts the processors that have been stopped. If the system cannot communicate with the operator console, it puts itself in restartable wait state X'114'. In earlier releases, the system displays the message for five seconds then automatically restarts all processors that were stopped.

Requiring a response improves system integrity. It ensures that the operator is aware of the lost resource and allows the operator options for recovery. The operator might, for example, want to re-IPL instead of restart processors that depended on the update being completed.

Numbering Conventions for Processor Complexes

Figure 4-2 on page 4-9 illustrates the different numbering conventions used for central processing units (CPs) in the IBM 3084 and the IBM 3090 model 400 processor complexes. You must be sure the CP numbers given in commands match the numbering system of your processor complex and your actual resource configuration.



Key:
CP = central processor
CHP = channel path
ESE = extended storage element
SE = storage element
VF = vector facility

Figure 4-2. Numbering Conventions for Processor Complexes

Summary of New, Updated, or Deleted Commands

Figure 4-3 summarizes the commands that are new, updated, or deleted in Release 2.1.x. Because updates to these and other commands may have occurred for Release 2.2.x, be sure to read the corresponding chart in the *MVS/XA Conversion Notebook, Volume 2*. Most of the updates are compatible (that is, commands that specify existing parameters, keywords, or options can be entered the same way in MVS/370 and MVS/XA). Exceptions are:

- TRACE (except TRACE STATUS)
- MODE E1, E2, E3, and E4
- VARY CHP, CPU, STOR, and PATH
- REPLY id, ASID in response to a DUMP command

These commands must be specified differently in MVS/XA.

The syntax of the following commands is the same. However, they produce different output in MVS/XA:

- DISPLAY M=DEV (With Release 2.1.3 the syntax of this command changes as well. See Figure 4-3 on page 4-11.)

- DISPLAY MPF

- REPLY SDATA=(NUC) or (TRT) in response to a DUMP command

System Commands describes the syntax of the commands and how to use them.

Command	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
CANCEL	x							Using the new A keyword, operators can terminate specified jobs, time-sharing users, or started processes that do not have unique names or have not yet been assigned job names. The A keyword identifies the task to be cancelled. If two tasks with the same name are both active when the operator issues a CANCEL command, the MVS/XA system rejects the command. The operator can then use A to specify the ASID of the task to be cancelled and reissue the command. In MVS/370, the system terminates the first task found. Operators can also use the A keyword to cancel partially-initiated tasks that have not yet been assigned names. The DISPLAY A,L command identifies those tasks as 'STARTING' or '*LOGON*.' The operator can cancel them by issuing a CANCEL command that specifies the A keyword and a job name of 'STARTING' or a user ID of '*LOGON*.'
CHNGDUMP	x							Following are new or changed options for SYSABEND, SYSMDUMP, SYSUDUMP, and SVC dumps: ALLVNUC, ALLNUC, NOSYM, NUC, SUM, TRT, and SUBTASKS. Figure 6-1 in Chapter 6 describes each of the options.
CONFIG	x							A new reconfiguration command that: - Physically and logically reconfigures channel paths, processors, and storage. The CONFIG CPU, CONFIG CHP, and CONFIG STOR commands, respectively, replace the MVS/370 VARY CPU, VARY CH, and VARY STOR commands. - Allows the operator to reconfigure the system as specified in a CONFIGxx PARMLIB member. The CONFIG MEMBER command requests this function. Note that the CONFIG MEMBER command does not reconfigure devices or DASD volumes. - Allows the operator to select elements to be reconfigured from a display of elements that are in the current configuration or that can be brought online. The operator issues either CONFIG ONLINE or CONFIG OFFLINE to request this function. - Allows the operator to determine the online or offline status of all available processors, channel paths, and storage in the configuration.
					x			The VF keyword on the CONFIG command allows the operator to reconfigure one or more Vector Facilities attached to online processors. The VFON and VFOFF operands on the CONFIG CPU(x), ONLINE command also allow the operator to reconfigure Vector Facilities.
							x	Using the ESTOR keyword you can reconfigure extended storage elements online or offline. The CONFIG ONLINE or CONFIG OFFLINE commands result in a new display that includes side information for the IBM 3090 model 400. **Note.** When using the CPU keyword the numbering convention for the IBM 3090 model 400 processors differs from the convention used for numbering the processors in the IBM 3084 processor complex. Use the DISPLAY M=SIDE command to see which processors are on which side of a partitioned processor complex.
CONTROL M			x					A new keyword, UEXIT, activates and deactivates the IEAVMXIT user exit. If IEAVMXIT is in the LNKLST concatenation at IPL time, the system automatically activates it. Thereafter, you can use the UEXIT keyword to control IEAVMXIT's status. IEAVMXIT is new in Release 2.1.2. See "New WTO/WTOR User Exits" in Chapter 5 for more information.

Figure 4-3 (Part 1 of 8). Summary of New, Updated, or Deleted Commands

Command	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
CONTROL S		x						Two new keywords: - L=cc specifies which console the CONTROL S command is to affect. In earlier releases, you can change the console specifications for only the console on which the CONTROL command is issued. - MFORM specifies the form in which messages are to be displayed. The choices are: (1) the message text alone, (2) the message text with the issuer's job name or the job ID, or (3) the message text with either the issuer's job name or the job ID and a timestamp.
CONTROL V		x						When processing a CONTROL V command that switches a console between full-capability and message stream modes, the system reestablishes the console specifications that were in effect the last time the console was in that mode during the IPL. Earlier releases reestablish the console specifications set during system generation.
			x					A new LEVEL keyword specifies which type of messages a particular console will accept. You can use LEVEL in addition to routing codes to further control message traffic. The options on LEVEL are: ALL - All messages routed to the console. ALL is the default. R - WTORs. I - Immediate action messages (descriptor codes 1 and 2). CE - Critical eventual action messages (descriptor code 11). E - Eventual action messages (descriptor code 3). IN - Informational messages, excluding broadcast and action messages. NB - No broadcast messages, regardless of the descriptor code. UNCOND is also an option on LEVEL. It indicates the system is to perform the LEVEL request unconditionally, even if some broadcast or informational messages are sent only to the hardcopy log as a result. (The system sends WTORs and action messages that are suppressed from all consoles to the master console as well as to the hardcopy log.) If you specify LEVEL=UNCOND and messages are sent to hardcopy only, you receive a warning message. To identify which messages are sent only to hardcopy, use a DISPLAY CONSOLES command with the HCONLY keyword specified. If you omit the UNCOND option on a command that would cause some messages to go to hardcopy only, the system rejects the command.
DISPLAY A					x			When you give the DISPLAY A, A (display system activity, all) command, you may see a *VF* in the AFF, processor affinity, field. This means that the job requires a processor with the Vector Facility and is swapped out waiting for one to become available.

Figure 4-3 (Part 2 of 8). Summary of New, Updated, or Deleted Commands

Command	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
DISPLAY CONSOLES			x					Several new keywords request status information about specific consoles. They are useful for limiting the display to information that is pertinent. Previously, DISPLAY CONSOLES always produced the status of all consoles in the system. The new keywords and the consoles they specify are: ACTIVE - Active consoles, including the master console and the hardcopy console. ACTIVE is the default. NACTIVE - Consoles that are not active. SS - Consoles that subsystems can use. CN(xx) - Consoles whose IDs are listed on CN. U(aaa) - Consoles identified by the device numbers on U. ROUT(rr) - Consoles that receive messages with the routing codes specified on ROUT. BACKLOG - Consoles with a message backlog. MASTER - The master console and any pseudo-master consoles. * - The console from which the DISPLAY command is issued. LIST - All consoles defined to the system. Thus, the output from LIST is equivalent to the output from previous DISPLAY CONSOLES commands. You can now route the DISPLAY CONSOLES command. A new L keyword specifies the display area, console, or both where the system is to display the output. The HCONLY keyword is also new. It requests information about messages that are recorded only on the hardcopy log and not sent to any console. The display lists routing codes not assigned to any console. If any broadcast messages are being sent to hardcopy only, the display also includes the word BROADCAST. The HCONLY keyword is introduced in Release 2.1.2 because of changes to the CONTROL V command. You can now use CONTROL V to specify which type of messages a console does or does not accept. Thus, it is possible that some broadcast and informational messages are routed to hardcopy only. For more detail, see the entry for the CONTROL V command.
DISPLAY DUMP	x							Two new keywords, TITLE and ERRDATA, display information stored in the header record of DASD SYS1.DUMP data sets that are full. The TITLE keyword lists the dump titles; the ERRDATA keyword displays error data from the dump header. TITLE and ERRDATA are valid options for displaying DASD dump data set information only. You can specify the DSN operand with the TITLE and ERRDATA keywords to restrict the display to a specific group of DASD dump data sets. The OPTIONS parameter supports the new SYSABEND, SYSUDUMP, SYSMDUMP, and SVC dump keywords. Figure 6-1 describes the new keywords. The information displayed when the STATUS operand is specified has changed. The display lists the full and available dump data sets grouped by device (DASD and tape). It does not provide any titles. Users must specify the TITLE operand to display dump titles. Previously, the display gave the status (full or available) of each dump data set on a separate line and included the titles of full dump data sets.

Figure 4-3 (Part 3 of 8). Summary of New, Updated, or Deleted Commands

Command	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
DISPLAY GRS			x					Release 2.1.2 provides new keywords for displaying additional global resource serialization information: - CONTENTION displays information about all resources for which at least one task is waiting. For a description of the information shown, see the RES keyword entry. - RES displays information about every resource that is currently allocated. The display includes the resource's rname, qname, and scope, and the following information about each task that owns or is waiting for the resource: the sysname, jobname, ASID, TCB address, type of request (shared or exclusive), and the task's status (owner or waiting). - RNL displays one or more RNLs, depending on which options RNL specifies. - HEX displays CONTENTION, RES, and RNL information in hexadecimal as well as regular format. - ALL displays the information that CONTENTION, RNL, SYSTEM, and LINK request. (SYSTEM and LINK are old keywords.) If you specify DISPLAY GRS with no keywords, you see SYSTEM and LINK information, which is consistent with earlier releases. *Note:* You can display CONTENTION, RES, and ALL information even if the system is not part of an active global resource serialization complex.
DISPLAY M	x							New and changed parameters: - **ddd** requests the online or offline status of all channel paths to device **ddd**. - **nn** requests the status of each device on channel path **nn**. - CHP requests the status of all channel paths in the system. - DEV displays the number of online channel paths to each device. In MVS/370, DEV displays the online or offline status of all devices and the channel sets to which the online devices are connected.
				x				The CHP and CPU keywords now allow operands. The DEV keyword format is updated to DISPLAY M=DEV(xxx) to match the format of the CONFIG keyword and the DEV output has been enhanced. The new ESTOR keyword displays the status of extended storage frames. See *System Commands* for complete descriptions of these updates.
					x			The CPU keyword gives a report on Vector Facility status.
							x	Message ID IEE490I has been replaced with message ID IEE174I for all displays produced by DISPLAY M commands. The SIDE and I/O keywords result in the same display which supports the IBM 3084 and the the IBM 3090 model 400 environments. The display is produced by the new message IEE174I and is described in detail in *System Messages Volume 2*. The SIDE display continues to be a part of the default DISPLAY M processing but the I/O display does not since it produces the same information.
DISPLAY MPF	x							The display now also includes the color and highlighting defaults for 3279 MCS consoles.
			x					DISPLAY MPF displays additional information in Release 2.1.2. It has two new keywords for limiting the output, MSG and COLOR. MSG displays: - The IDs of messages MPF is to suppress - The IDs of action messages that the action message retention facility does not retain - Which WTO/WTOR user exits are associated with which messages - Whether or not the general WTO/WTOR exit (IEAVMXIT) is active COLOR displays the color, intensity, and highlighting options in effect for 3279 consoles. If you specify no keywords, you see all of the information.

Figure 4-3 (Part 4 of 8). Summary of New, Updated, or Deleted Commands

Command	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
DUMP	x							The syntax of the DUMP command is the same; the content of the reply to the command is different. The NUC option on the SDATA keyword requests only the non-page-protected DAT-on section of the nucleus. Previously, it requested the entire nucleus. ALLNUC, a new option on the SDATA parameter, requests a dump of the DAT-off nucleus and the entire DAT-on nucleus, including the page-protected section. Users must specify the ASID list in hexadecimal rather than decimal values. Thus, the ASID specifications in the dump request are consistent with the message describing its completion and the internal description of the dump contents.
DUMPDS	x							Using the DUMPDS command operators can: - Add or delete SYS1.DUMP data sets after IPL/NIP time without having to re-IPL. Before adding a data set, it must be allocated and cataloged. In MVS/370, installations can only add and delete dump data sets at IPL/NIP time. Be aware that if the DUMPSRV address space terminates, you must again add dump data sets that were added using the DUMPDS command. When DUMPSRV restarts, the system adds the dump data sets that were added at IPL/NIP time. - Clear SYS1.DUMP data sets on DASD or tape. The DUMPDS CLEAR command avoids having to either run a utility job (AMDPRDMP or IEBGENER) or reset, load, and ready a tape drive to clear a DASD dump data set. Operators can issue the DUMPDS command only from consoles that have system authority.
FORCE	x							Using new A and ARM options on the FORCE command, operators can terminate specified jobs, time-sharing users, or started processes that: - Do not have unique names - Have not yet been assigned job names. - Are not eligible for cancellation via the CANCEL command. If two tasks with the same name are both active when the operator issues a FORCE command, the MVS/XA system rejects the command. The operator can then use A to specify the ASID of the task to be cancelled and reissue the command. In MVS/370, the system terminates the first task found. Operators can also use the A keyword to cancel partially-initiated tasks that have not yet been assigned names. The DISPLAY A,L command identifies those tasks as 'STARTING' or '*LOGON*.' The operator can cancel them by issuing a FORCE command that specifies the A keyword and a job name of 'STARTING' or a user ID of '*LOGON*.' The ARM keyword requests that the specified job, time-sharing user, or started process be terminated, even if it is not eligible for cancellation via the CANCEL command. Operators can use ARM to cancel any address space except one that is not eligible for termination (ASCBNOMT = 1).
MODE	x							ENABLE, E1, E2, E3, and E4 are no longer valid parameters and cause the MODE command to be rejected.
					x			When machine check interruptions caused by a Vector Facility reach a certain threshold, the Vector Facility is disconnected. The VS keyword on the MODE command allows the operator to change the default threshold.
MODIFY		x						New keywords, LLA, REFRESH, cause the system to rebuild the LNKLST lookaside (LLA) directory. The LLA directory is new in Release 2.1.1. See "Using a New Directory for LNKLST Data Sets" in Chapter 8 for more information.

Figure 4-3 (Part 5 of 8). Summary of New, Updated, or Deleted Commands

Command	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
MONITOR			x					You can now route the MONITOR command. In earlier releases, the output is always displayed at the console from which MONITOR is issued. To route MONITOR, either: - Include the L=cc keyword on MONITOR. L=cc specifies the ID of the console on which the output is to be displayed. - Issue an MSGRT command that specifies the new MN keyword.
MSGRT	x							A new keyword, CF, specifies to which MCS console the system is to route CONFIG commands.
		x						You can now issue the MSGRT K command from display, as well as non-display, consoles. Earlier releases restrict its use to non-display consoles only.
			x					A new MN keyword routes MONITOR and STOPMN commands. MN specifies the console to which the commands are to be routed. Neither of those commands can be routed in earlier releases.
SET		x						DAE=xx is a new keyword that specifies which ADYSETxx PARMLIB member the system is to process. ADYSETxx contains options for controlling dump analysis and elimination (DAE). Issuing SET DAE=xx causes the system to begin using the options in the specified member. See "Dump Analysis and Elimination (DAE)" in Chapter 6 for more information. You can now use SET SMF to restart SMF after it is terminated. Because SMF runs in its own address space, you no longer need to perform an IPL to restart it.
SLIP MOD			x					You can now enable or disable several SLIP traps with one command. Release 2.1.2 allows asterisks in place of any or all of the four characters of the ID keyword. The system enables or disables all traps having identifiers that match the characters you do specify. For example, SLIP MOD,ENABLE,ID=0*** causes the system to enable all SLIP traps that have identifiers beginning with 0.
SLIP SET	x							New options and keywords in Release 2.1.0: - NOSYSA, NOSYSM, NOSYSU, and NOSVCD are new options on the ACTION keyword. They suppress respectively SYSABEND, SYSMDUMP, SYSUDUMP, and SVC dumps for specified abend conditions. See "Suppressing Dumps" in Chapter 6 for more information. - ALLNUC on the SDATA keyword requests the entire nucleus (both the DAT-on and DAT-off sections). Indirect addresses on SLIP command keywords (DATA, LIST, SUMLIST, and TRDATA) can be 31-bit values. To indicate that an indirect address is to be treated as a 31-bit value, use a '?' instead of a '%' as the indirect address indicator. When a keyword specifies '%', the system treats the indirect address as a 24-bit value.
		x						New options and keywords in Release 2.1.1: - NUCMOD specifies a nucleus module or an address range within a nucleus module. When specified on an error event trap, NUCMOD indicates the range within which the error must occur. On IF or SB PER traps, it establishes the range of addresses to be monitored. On SA PER traps, NUCMOD specifies boundaries for the instruction causing the storage alteration. - NOSUP, a new parameter on the ACTION keyword of error event traps, indicates that the system is to take dumps for the trapped event, regardless of any attempts to suppress the dumps. Thus, NOSUP overrides dump suppression via dump analysis and elimination (DAE) or the ABDUMP pre-dump exit routine. - AND (&) or OR (|) on the DATA keyword logically compare triplets of target locations, operators, and values. You can specify AND and OR together with any number of triplets. You can also group triplets using parentheses. If you specify no logical operator, AND is the default, which is consistent with earlier releases. In earlier releases, AND is the implied logical operator.

Figure 4-3 (Part 6 of 8). Summary of New, Updated, or Deleted Commands

Command	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
SLIP SET (continued)								- REASON, a new keyword on error event traps, specifies a reason code for filtering errors. It is valid only with the COMP (completion code) keyword.
			x					New keywords and options in Release 2.1.2 are: - RECORD, a new option on the ACTION keyword of error event traps causes recovery routines to record the trapped error in SYS1.LOGREC. - STRACE, a new option on the ACTION keyword of PER traps, causes the system to write a system trace table entry for each trapped event. Thus, you can use PER to isolate problems without stopping the system when the program event occurs. - PVTMOD is now valid on all PER and non-PER traps. Previously, it was an option only on non-PER and storage alteration PER traps. Thus, for the first time you can monitor instruction fetch and successful branch events in the private area. - PVTEP, LPAEP, and NUCEP specify entry points in modules that reside in the private area, LPA, and nucleus, respectively. They cause monitoring to begin at the address associated with the entry point or at the specified offset from that address. PVTEP and LPAEP are particularly useful for monitoring the section of a module that begins at an alternate entry point. NUCEP is equivalent to NUCMOD. You can now monitor events in modules whose names end in x'C0' (SVC load modules). Because x'C0' is not alphanumeric, the system does not accept it on the LPAMOD, LPAEP, NUCMOD, NUCEP, PVTMOD, or PVTEP keywords. However, you can now use an asterisk in place of x'C0'. The system interprets the asterisk as x'C0'.
START	x							A new keyword, SUB, directs the JCL for a started task to the internal reader of a secondary JES or to the master subsystem. By routing started tasks to a secondary JES or to the master subsystem, operators can: - Run started tasks independently of the primary JES. - Start certain tasks before the primary JES initialization is completed. See *SPL: System Modifications* for more information.
		x						The procedure name, LLA, is used as a keyword to start the LLA procedure, which in turn starts the LNKLST lookaside (LLA) function. The LLA function builds and maintains a new directory of modules in the LNKLST concatenation. BLDL then searches that directory instead of the PDS directories for LNKLST modules. For more information about the LLA function, see "Using a New Directory for LNKLST Data Sets" in Chapter 8.
						x		The procedure name, AVM, is used as a keyword to start the AVM procedure, which in turn starts availability manager address space. This is the first step in using XRF. For more details see *XRF Planning*.
STOP		x						The procedure name, LLA, is used as a keyword to stop the LNKLST lookaside (LLA) function. The system then searches PDS directories instead of the LLA directory to locate modules in the LNKLST concatenation. For more information, see "Using a New Directory for LNKLST Data Sets" in Chapter 8.
STOPMN			x					You can now use STOPMN to suppress MONITOR command output from any console except the one from which STOPMN is issued. A new keyword, L=cc, identifies the console on which the output is to be suppressed.

Figure 4-3 (Part 7 of 8). Summary of New, Updated, or Deleted Commands

Command	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
TRACE	x							The syntax of the TRACE command has changed. Except for TRACE STATUS, MVS/370 TRACE commands do not work in MVS/XA. In addition, the TRACE command controls system tracing differently: - MVS/XA continues system tracing after system initialization, unless an installation requests that tracing be stopped. Thus, no TRACE command is required in the COMMNDxx PARMLIB member to request system tracing. In MVS/370, to continue system tracing after system initialization, you have to issue a TRACE ON command. - The MVS/XA trace facility performs branch tracing, address space tracing, or explicit tracing. ("Changes to the System Trace Facility" in Chapter 5 lists the system events in each category.) A new TRACE operand, BR, allows installations to start and stop branch tracing independently of address space and explicit tracing. Options are: --Explicit and ASID tracing on, branch tracing off --All tracing on --All tracing off The system can perform branch tracing only when the other trace options are active. - TRACE can start or stop system tracing or change the TRACE options at any time after system initialization is completed. To start system tracing in MVS/370, you have to issue a TRACE ON command before the system starts JES2 or JES3. Also, in MVS/370 you cannot use the TRACE command to stop system tracing after subsystem initialization is completed. - TRACE allows you to dynamically change the size of the system trace table using a new option on the ST keyword. The default size is 16 K per processor. In MVS/370, the trace table size is fixed at IPL time. The master trace (MT) and display trace status (STATUS) functions of the TRACE command are not changed. However, you must use the new syntax of the TRACE command to start or stop master trace, as well as system trace. The order in which you specify the parameters is changed. *System Commands* describes the new syntax.
VARY CPU, VARY CH, VARY STOR	x							The VARY CPU, VARY CH, and VARY STOR commands are deleted. The new CONFIG CPU, CONFIG CHP, and CONFIG STOR commands perform equivalent functions in MVS/XA. See the CONFIG entry in this table.
VARY devnum, ONLINE			x					In JES2 environments the SHR keyword allows more than one processor to share the 3480 magnetic tape subsystem.
VARY PATH	x							The syntax is changed to support the new I/O architecture. MVS/370 VARY PATH commands do not work in MVS/XA. Also, the VARY PATH,OFFLINE command might take longer to process. The system will not vary a device path offline until I/O activity to the target device has completed. If I/O is not completed after 150 seconds, the system issues message IEE717D, which gives the operator a chance to cancel the command.

Figure 4-3 (Part 8 of 8). Summary of New, Updated, or Deleted Commands

Chapter 5. System Modifications

This chapter contains information related to modifying the system. The topics it includes are:

Dynamically Updating the SVC Table

The SVCUPDTE macro, provided in Release 2.1.0, allows authorized programs to dynamically update the SVC table. These updates take effect without an IPL.

Updating SYSTEMS Exclusion RNLs

Beginning with Release 2.1.1, if your system is part of a global resource serialization complex, you might have to add the resource name for the SYS1.DAE data set (SYSDSN, SYS1.DAE) to the SYSTEMS exclusion RNLs of other systems in the complex. Because SYS1.DAE cannot be shared among systems, the SYSTEMS exclusion RNL shipped in Release 2.1.1 includes an entry for SYS1.DAE. Global resource serialization requires that all systems in a complex have identical RNLs. Therefore, you need to add the same entry to the SYSTEMS exclusion RNLs of any system in the complex that is not at Release 2.1.1 or a later level.

Serializing VSAM Data Sets

If your installation has a global resource serialization complex that includes both systems with MVS/XA DFP 1.1 or later, MVS/XA DFP 2.1.0 or later, or MVS/370 DFP installed and systems with neither, you need to take certain actions to ensure that VSAM data sets are serialized correctly. These releases of DFP include an updated VSAM. The ENQs that the new VSAM OPEN processing issues to enforce VSAM cross-region share options 1 and 2 are different. The scopes of the ENQs are changed to SYSTEMS, and their rnames (minor names) include catalog names. Only the qnames (major names) remain the same.

A catalog name is system-independent information. Therefore, if all systems in the complex include the new level of VSAM and all systems accessing the data set belong to the complex, you can use the VSAM ENQ to serialize access to the data set as a global resource. The scope of SYSTEMS causes global resource serialization to treat the data set as a global resource by default.

The old level of VSAM uses ENQs with scopes of SYSTEM and rnames that include catalog ACB addresses. Because catalog ACB addresses vary from system to system, you cannot use these ENQs to serialize access to data sets as global resources. The scope of SYSTEM causes global resource serialization to treat the data set as a local resource by default.

The following figure summarizes the differences in the ENQs:

	Old VSAM ENQs	**New VSAM ENQs**
QNAME	SYSVSAM	SYSVSAM
RNAME	system-dependent	system-independent
SCOPE	SYSTEM (local resource)	SYSTEMS (global resource)

Because of the differences, systems in complexes that include both levels of VSAM cannot share the VSAM data sets globally. Therefore, place a generic entry for SYSVSAM in the SYSTEMS exclusion RNL of each system. Also ensure that the SYSTEM inclusion RNLs do NOT include an entry for SYSVSAM.

After all systems in the complex have the new level of VSAM installed, remove the SYSVSAM entry from the SYSTEMS exclusion RNLs. The VSAM OPEN processing then enforces share options 1 and 2 as follows:

- If a data set assigned share option 1 is opened for output, no other user in the complex can open it for output or input.
- If a data set assigned share option 2 is opened for output, other users in the complex can open it for input but not output.

Limiting User Region Size Using IEFUSI Instead of IEALIMIT

Installations can now use the SMF step initiation exit (IEFUSI) to limit the sizes of user regions above and below 16 megabytes. The methods available in MVS/370 continue to work in MVS/XA:

- Specifying the REGION parameter on JCL statements
- Assigning default values through JES2 or JES3
- Using the IEALIMIT installation exit

However, using IEFUSI has the following advantages:

- IEFUSI is a separate load module in the LPA. IEALIMIT must reside in the nucleus. Thus, you must link edit the nucleus every time you replace IEALIMIT with a new version.

- IEFUSI users can readily obtain information required to set a region size and region limit. IEALIMIT must scan system control blocks to gather that information. Thus, IEFUSI is easier to write and less susceptible to system changes.

- IEALIMIT requires that the local lock be held and, therefore, cannot issue SVCs. IEFUSI has neither of those restrictions.

- IEFUSI can control the region size and region limit of both the area above and the area below 16 megabytes. IEALIMIT can set values for the area below 16 megabytes only; VSM uses defaults defined in the code for the area above 16 megabytes.

- IEFUSI can lower the extended limit and extended region size from the 32 megabyte minimum when the fix for APAR OY04728 is installed.

To indicate that VSM is to use IEFUSI instead of IEALIMIT for controlling the user region area, you must set a flag in the IEFUSI parameter list. VSM then bypasses the IEALIMIT exit and limits the user area as IEFUSI requests. IEFUSI does not have to set new limits. It can, for example, request that VSM set limits identical to the IEALIMIT defaults.

SPL: User Exits describes how to use IEALIMIT. *SPL: SMF* describes how to use IEFUSI. *SPL: System Modifications* provides general information on limiting the user region.

Using the REGION Parameter

VSM changes in Release 2.1.2 make it easier for a job to obtain an extended region size greater than 32 megabytes:

- You can now specify values greater than 16 megabytes on the JCL REGION parameter. The values can be expressed in K or Mb units and can be as high as 2096128 K or 2047 Mb.

- VSM now uses the REGION parameter (if nonzero) to calculate the extended region size and the limit for GETMAIN requests above 16 megabytes. VSM sets both to the smaller of: (1) the size of the extended private area, or (2) the

REGION parameter value or 32 megabytes, whichever is greater. In earlier releases, VSM sets both to 32 megabytes.

If the REGION parameter is greater than 16 megabytes, the only limits on GETMAIN requests below 16 megabytes and the region size below 16 megabytes are the limits that IEALIMIT or IEFUSI set, or the size of the private area. *SPL: System Modifications* contains more information about limiting the user region size. Also, as of Release 2.1.2, jobs that use the OS/VS1 form of the JCL REGION parameter (with two arguments: REGION=(xxK,yyyK)) abnormally terminate. Previously, the system ignored this form of the REGION parameter when it was used by MVS/XA jobs.

Bypassing the Storage Availability Check Before a Job Executes

If Release 2.1.2 or a subsequent release is installed, you can control whether or not the virtual storage manager (VSM) checks that the amount of storage requested on the REGION parameter is available before permitting a job to execute. (Earlier levels of MVS/XA did not allow you to control this check.) The change is intended to prevent jobs from failing simply because programmers specify REGION values that are unnecessarily large.

Even though you can control whether or not VSM makes the availability check, it is not required that you do so. If you choose not to exercise any control, VSM processing, by default, will be the same as in pre-Release 2.1.2 versions of MVS/XA. The following default actions occur:

- If the value specified on the REGION parameter is less than 16 megabytes, VSM ensures that there is at least that much storage below 16 megabytes.
- If the value specified on the REGION parameter is greater than 16 megabytes, VSM does not make any storage availability check.

To change the default, use IEFUSI. Bits 1 and 2 in the first word of the VSM parameter list passed to IEFUSI control checking. Bit 1 indicates whether VSM is to check for available storage below 16 megabytes. Bit 2 controls checking above 16 megabytes. See *SPL: System Modifications* and *SPL: SMF* for more detail.

Changing the Hot I/O Threshold and Recovery Actions

A new HOTIO statement in the Release 2.1.2 IECIOSxx PARMLIB member makes it easier to change: (1) the threshold IOS uses for detecting hot I/O conditions, and (2) the recovery actions IOS performs when it detects a hot I/O condition. Module IOSRHIDT contains the threshold and recovery actions. It replaces the MVS/370 IECVHIDT module. Previously, you had to use the AMASPZAP service aid to update IOSRHIDT (or IECVHIDT).

The first release of MVS/SP Version 2 changes hot I/O processing. "Processing Hot I/O Interrupts" in Chapter 4 briefly describes the changes. *SPL: System Modifications* contains more detail. *SPL: Initialization and Tuning* describes how to write the HOTIO statement.

Pre-dump Exits

Installations can now provide exit routines that get control before the ABDUMP routine takes a dump. The exits can analyze the requested dump and either:

- Continue with the dump as requested
- Modify the dump options and continue with the dump
- Terminate the dump request.

IBM supplies a load module, IEAVTABX, that contains blank entries for the pre-dump exit routine names. *SPL: System Modifications* describes pre-dump exits. *SPL: User Exits* provides coding information.

Post-dump Exits

Installations can now provide exit routines that get control after each SYSMDUMP and SVC dump is taken. The post-dump exit routines can examine dumps in dump data sets, evaluate the dump and the dump symptoms, and take appropriate action (for example, tell the operator to clear the dump data set using the new DUMPDS command or to start a PROC to offload the dump data set). IBM supplies a new load module, IEAVTSEL, that contains an SDUMP exit list with blank entries.

SPL: System Modifications describes post-dump exits in more detail. *SPL: User Exits* provides coding information.

RMF Exits

RMF exits require modification. RMF invokes all user exits in 31-bit addressing mode and expects return in that mode. In addition, most RMF user exits need to access control blocks that have been moved to virtual storage above 16 megabytes. The *RMF Reference and User's Guide* explains how to write RMF exits.

JES2 User Exits and Interfaces

A JES2 user exit that uses the downward incompatible macros listed in "Handling Downward Incompatible Macros" on page 9-7 needs to ensure that the exit (the part of it that uses JES2 interfaces) contains the MVS/370 macro expansions. See *MVS/XA JES2 User Modifications and Macros* for details on which macro expansions you need and how to obtain them.

JES3 Dynamic Support Programs (DSPs) and User Exits

Beginning with Release 2.1.5, JES3 invokes DSPs and most user exits in 31-bit addressing mode and expects return in that mode. Thus, JES3 user-written programs can usually reside anywhere in virtual storage and use JES3 interfaces while in 31-bit addressing mode. See the *JES3 Conversion Notebook* for details on user exits not invoked in 31-bit addressing mode. If your user exit needs the downward incompatible macros described in "Handling Downward Incompatible Macros" on page 9-7, see *MVS/XA JES3 User Modifications and Macros* for programming details.

PRDMP/IPCS Exit Control Table (ECT) Modifications

For Release 2.1.x., the system uses some previously-empty ECT entries for new PRDMP exits. If your installation added ECT entries, you must add them to the new ECT at different offsets. The ECT is in module AMDPRECT. See the *MVS/XA Conversion Notebook, Volume 2* for a description of Release 2.2.x changes to the ECT and AMDPRECT.

PRDMP Exits

You need to change PRDMP exit routines that use the print service and supply their own output buffers. In MVS/XA, the print service routine expects a 132-byte output buffer. It prints 132 bytes, beginning at the output buffer address specified in the ADPLBUF field of the exit parameter list (BLSABDPL). In MVS/370, the output buffer is 121 bytes, but only 120 bytes are printed.

PRDMP exits that use the PRDMP-supplied output buffer continue to work unchanged in MVS/XA. The buffer is set to blanks after each use. However, if you modify the ADPLBUF field to point to a 121-byte output buffer, either:

- Each printed line will contain 12 bytes of unexpected data.
- The PRDMP exit will fail with an x'0C4' ABEND.

PRDMP Header Exits

The Release 2.1.1 level of PRDMP allows a new type of user exit, header exits. Using header exits you can add information to the title pages of dumps. PRDMP calls header exits when processing title pages.

Dump analysis and elimination (DAE) supplies one header exit, ADYHDFMT, which formats and prints DAE symptom data. You can supply additional header exits, but do not change the ECT (exit control table) entry for ADYHDFMT (entry 21). PRDMP also calls that entry when processing the new DAEDATA PRDMP verb.

SMF Exits

You might have to modify two SMF exits, IEFU29 and IEFU84, beginning with Release 2.1.1. IEFU29 and sometimes IEFU84 run in the new SMF address space instead of the master scheduler address space. (IEFU84 runs in the SMF address space when SMF calls it during system initialization to write record types 0, 8, 19, 22, and 90.) If either exit requires data located in the private area of the master scheduler address space, you need to change the exit. Use cross memory instructions (SSAR, MVCP, and MVCS) to move data between the two address spaces.

Other SMF exits continue to run unchanged in the same address spaces as in previous releases of MVS/XA. You can, however, write SMF exits that run in 31-bit addressing mode, reside above 16 megabytes, or address data above 16 megabytes. Exits that run in 31-bit addressing mode must return control to SMF using a BSM instruction. In addition, SMF records that IEFU83 or IEFU84

passes to SMF must reside below 16 megabytes. See *SPL: 31-bit Addressing* for help in writing programs that use 31-bit addresses.

Entering IEECVXIT into the Control Program

Beginning with Release 2.1.2 IEECVXIT, the IBM-supplied WTO/WTOR exit routine, is no longer a part of module IGC0003E. IGC0003E has moved above the 16 megabyte virtual storage address, and IEECVXIT now resides in its own load module, also called IEECVXIT, as part of SYS1.LPALIB. Use the linkage editor to replace the IBM-supplied dummy routine with your own routine. The *SPL: User Exits* provides further information on IEECVXIT.

New WTO/WTOR User Exits

Installations with Release 2.1.2 or subsequent releases installed can use new WTO/WTOR exits to modify message processing. The new exits are in addition to IEECVXIT. Unlike IEECVXIT, they can modify processing for any message. They can also alter processing in more ways than IEECVXIT can. The new exits can:

- Alter routing and descriptor codes.
- Change the message text.
- Change the console on which the message is displayed.
- Queue messages to a particular active console; queue them unconditionally to any console, regardless of whether it is active; or queue messages by routing codes only.
- Direct messages to the hardcopy log only, to consoles only, or to both the hardcopy log and consoles.
- Delete messages, except WTORs.
- Control whether or not messages are broadcast to active consoles.
- Override MPF suppression.
- Issue SVCs (for example, SVC 34 and SVC 35).
- Reply to or suppress WTORs.
- Control whether or not the action message retention facility retains a message.

You can find examples of coding these user exits in the *SPL: User Exits.*

The system invokes WTO/WTOR exits after IEECVXIT and MPF processing is completed and before calling the subsystem interface (SSI). It can invoke only one WTO/WTOR exit for each message processed. However, you can provide several exits and specify which the system is to call for particular messages. You can also write one general exit that the system invokes for all messages not

associated with a specific exit. You must name the general exit IEAVMXIT. Both the general and specific WTO/WTOR exits must execute in 31-bit addressing mode and reside in an authorized data set that is included in the LNKLST concatenation.

To associate a specific exit with a message or group of messages, include statements like the following in the MPFLSTxx PARMLIB member:

```
message ID   USEREXIT(name of exit routine)
```

The message ID can identify either a particular message or a class of messages, for example, IEF404I or IEF*. The system then calls the specified exit when processing those messages. (You can also put the new RETAIN and SUP keywords on the same statement as USEREXIT. See "New, Updated, or Deleted PARMLIB Members" in Chapter 2 for more information.) Use multiple MPFLSTxx members and the SET MPF=xx or SET MPF=NO command to control which specific exits are active.

If the system is to call IEAVMXIT when processing a message, you need not do anything except make that exit available and possibly activate it. If IEAVMXIT is in the LNKLIST concatenation at IPL time, the system automatically activates it. Thereafter, you can activate and deactivate it using a new UEXIT=Y|N keyword in the CONTROL M command.

New Services for Dump Processing Exits

Beginning with Release 2.1.2 two new services are provided for dump processing exits that are invoked from IPCS, PRDMP, and SNAP:

- Format model processor service
- Control block formatter service

Beginning with Release 2.1.2 two new services are provided for dump processing exits that are invoked from IPCS and PRDMP:

- ECT service
- Select ASID service

Beginning with Release 2.1.2 two new services are provided for dump processing exits that are only invoked from IPCS:

- GET symbol service
- EQUATE symbol service

In addition, as of Release 2.1.2 there is a new exit services router that serves as an interface between dump exits and new and existing exit services. However, both Release 2.1.2 and 2.1.3 continue to support existing interfaces between dump exits and the services available to them. Thus, existing dump exits will run unchanged in Releases 2.1.2 and 2.1.3.

Exit Services Router

Dump exits can use the exit services router to invoke the following services:

- Storage access service
- Print service
- Format model processor service
- Control block formatter service
- Index service (exits invoked by PRDMP only)
- ECT service
- GET symbol service (exits invoked by IPCS only)
- EQUATE symbol service (exits invoked by IPCS only)
- Select ASID service

The storage access, print, and index services are not new. The service router simply provides a new method of invoking them.

Dump exits can invoke any of the services listed by calling the exit service router and passing it three parameters:

- The address of the user exit parameter list, ABDPL, which is mapped by BLSABDPL.

- A service code indicating which service the router is to invoke.

- The address of the parameter list that the requested service expects (except when invoking the print or index service, which use no parameter list). ABDPL includes mappings of the parameter lists for each service except EQUATE and GET. The BLSRESSY macro maps the parameter lists for these two services.

Most services also require some additional information in fields of the ABDPL. *IPCS User's Guide and Reference* describes those requirements. It also describes in more detail how to invoke services using the exit services router.

Format Model Processor Service

The format model processor service formats and prints a control block or other data area using a format model. The exit routine supplies the model and the virtual address or buffer address of the storage to be formatted. The format model processor returns the formatted control block.

In addition to formatting control blocks, you can use format models and the format model processor to:

- Decode flag bytes
- Summarize dump data
- Present data in hexadecimal representation
- Display 2-dimensional arrays

The *IPCS User's Guide and Reference* describes each of these uses in more detail.

Using format models is an alternative to using format patterns. A format model consists of several BLSQMDEF and BLSQMFLD macros. The BLSQMDEF macro begins the model and describes the header. A series of BLSQMFLD

macros follow, one for each field to be formatted. A second BLSQMDEF macro denotes the end of the model. The models can be part of a program load module or separate load modules. *System Macros and Facilities* describes how to write BLSQMDEF and BLSQMFLD macros. The *IPCS User's Guide and Reference* shows how to write format models.

Format models are flexible. Programs that use models can call a user-supplied exit routine to examine or modify a formatted line before printing it. They can also vary which fields of the model are printed. Thus, programs can use the same model to format different levels of the control block. Models can format arrays within a control block. They are also independent of the line length, starting column, and column spacing. PRDMP, IPCS, or SNAP modules determine that information.

Control Block Formatter Service

The control block formatter service formats and prints a control block, given a control block acronym and the virtual or buffer address of the control block. The *IPCS User's Guide and Reference* (GC28-1297) gives a complete list of the control blocks you can format using this service.

Using the control block formatter has several advantages over using format patterns:

- You need to know nothing about the control block structure.
- You can invoke the service once and format the entire control block.
- If the control block format changes, only the control block model that the control block formatter service uses needs to be modified.
- You can request multiple levels of detail using only one control block model.

ECT Service

The ECT service searches the ECT (exit control table) for the requested exit, then links to it. This service allows one dump exit to invoke another.

GET Symbol Service

The GET symbol service returns a specified symbol from the IPCS symbol table. Thus, it performs the same function as the GET subcommand of IPCS and is meaningful only during IPCS processing. PRDMP and SNAP modules ignore requests for this service.

EQUATE Symbol Service

The EQUATE symbol service adds a symbol to the IPCS symbol table, as does the EQUATE subcommand of IPCS. Like the GET symbol service, it is meaningful only during IPCS processing. PRDMP and SNAP modules ignore requests for this service.

Select ASID Service

The select ASID service returns a list of pointers to ASCBs in a dump. The exit routine specifies which ASCB pointers are to be returned using one or more of the following keywords:

ALL	All ASCBs in the dump
CURRENT	ASCBs for address spaces that were active at the time the dump was taken
ERROR	ASCBs for all address spaces that terminated abnormally (ASCBCC≠0) or that contain TCBs representing tasks that completed abnormally (TCBCMP≠0 or TCBRTWA≠0)
TCBERROR	ASCBs for all address spaces that contain TCBs representing tasks that completed abnormally (TCBCMP≠0 or TCBRTWA≠0). TCBERROR specifies a subset of the ASCBs that ERROR selects.
ASIDLIST	ASCBs corresponding to the ASIDs listed on ASIDLIST.
JOBLIST	ASCBs corresponding to the jobs listed on JOBLIST.

SNAP dumps ignore requests for the ASID service.

Expanded (Extended) Storage Criteria

As of Release 2.1.3 your processor complex may include expanded storage as well as real and auxiliary storage. Expanded storage is an electronic extension of real storage provided to take advantage of the high speed of the processors and improve system responsiveness.

Expanded storage is also known as extended storage, and the terms are used interchangeably throughout the MVS/XA library. This *MVS/XA Conversion Notebook*, for example, uses the term extended storage in the discussion that follows and in other sections to refer to this form of storage.

Extended storage is most advantageous for processor complexes that include applications that depend upon extremely rapid system response. When the demand for real storage is high, the system transfers some code and data not currently in use to either extended or auxiliary storage. The system moves information that is most likely to be needed soon, that is most response sensitive, to extended storage and information that is not as response sensitive to auxiliary storage.

Transfer of information between extended and real storage occurs quickly because it takes place synchronously with processor operations, requiring no I/O. Transfer of information between auxiliary and real storage is slower; it requires asynchronous I/O processing.

There is no need to modify any MVS/XA system parameters to take advantage of extended storage. As of Release 2.1.3 there are fields in the IEAOPTxx member of SYS1.PARMLIB containing values that represent the priority of various categories of code and data. SRM storage management routines consider IBM-supplied defaults for these values when making recommendations to the real storage manager as to whether or not a page should be moved to extended storage.

However, an installation that chooses to modify the MVS/XA system and affect system responsiveness can change the values for each category by using IEAOPTxx SRM parameter keywords. *SPL: Initialization and Tuning* lists the keywords and explains how to use them.

Chapter 6. Problem Determination

This chapter includes information related to dumping services, trace facilities, and debugging.

The following topics describe differences in dump content and format:

Topics describing new ways of suppressing dumps include:

Print dump (PRDMP) changes are described in:

The topics below describe IPCS changes introduced in Release 2.1.2. Note that to use Release 2.1.2 IPCS dialogs, you must also have ISPF Version 2 installed.

"Using the MVS/XA Versions of IPCS and PRDMP on Other Systems" on page 6-18 describes how to obtain MVS/XA versions of IPCS and PRDMP on earlier systems.

Debugging considerations include:

New and Updated Dump Options

Figure 6-1 summarizes the new and updated dump options for user and system dumps that occurred during Release 2.1.x. It indicates whether the option is valid on the SNAP macro, the SDUMP macro, and/or the DUMP command. None of the updates are incompatible. However, the following MVS/370 options produce different dump data in MVS/XA: NUC, TRT, CB, SPLS, and SQA. Some options (for example, NUC and SUBPLST) are designed or updated to eliminate unnecessary dump data. Other options (such as SUM) improve the usefulness of dump data.

The new and updated dump options also occur in the following PARMLIB members: IEAABD00 (SYSABEND), IEADMP00 (SYSUDUMP), and IEADMR00 (SYSMDUMP). See Figure 2-3 on page 2-20 in Chapter 2.

Dump Option	SNAP	SDUMP	DUMP	Data Included in the Dump
ALLNUC (new)		x	x	The entire nucleus, both the DAT-on nucleus and the DAT-off nucleus. (NUC no longer requests the entire nucleus.) Although ALLNUC is an SDATA keyword, specifying SDATA=ALL does not cause the ALLNUC information to be dumped. You must explicitly specify the ALLNUC option to obtain the entire nucleus. Your installation might want to keep one copy of a dump of the page-protected nucleus to use with other dumps. To obtain a dump of only the nucleus, use a DUMP command with the ALLNUC, NOSQA, and NOSUM SDATA options specified.
ALLVNUC (new)	x			The entire DAT-on nucleus. Users cannot obtain the DAT-off nucleus in a formatted dump. However, ALLVNUC causes the system to dump both the DAT-on nucleus and the DAT-off nucleus in SYSMDUMPs. Although ALLVNUC is an SDATA option, specifying SDATA=ALL does not cause the ALLVNUC information to be dumped. When ALLVNUC is specified, dumps also include the PSA and the CVT. Unauthorized users receive only the section of the PSA that is not fetch-protected (locations 0 through 2 K minus 1). Authorized (key 0) users receive the entire PSA.
CB (changed)	x			In SYSABEND, SYSUDUMP, and SNAP dumps, a storage map that contains: - The starting storage address - The subpool number - The length of the storage allocated to the task - The storage key - Flag information pertaining to the storage and the owning TCB, if the storage is shared In MVS/370, the CB option causes formatted VSM control blocks to be dumped.
CSA and LSQA (changed)	x	x	x	The specified area below and above 16 Mb. No additional SDATA options are required to include extended virtual storage areas in a dump.
KEYLIST (new)		x		Areas in the subpools listed on the SUBPLIST keyword that have the specified key(s). KEYLIST is only valid when specified with SUBPLST. It allows users to obtain a subset of the specified subpool storage.

Figure 6-1 (Part 1 of 2). New, Updated, or Deleted Dump Options

Dump Option	SNAP	SDUMP	DUMP	Data Included in the Dump
NUC (changed)	x	x	x	The DAT-on, non-page-protected section of the nucleus, the PSA, and the CVT. Authorized (key 0) users receive the entire PSA. Unauthorized users receive only the section of the PSA that is not fetch-protected (locations 0 through 2 K minus 1). Users that need the page-protected section must specify either the ALLVNUC or ALLNUC option. In MVS/370, NUC requests the entire nucleus. The change in NUC output is designed to eliminate from dumps those areas of the nucleus that are not expected to change (the page-protected areas). Those areas are normally not required to debug problems. The change was made in such a way that most installations can suppress the page-protected areas without having to respecify dump options.
SPLS (changed)	x			Subpool storage printed in ascending address order, instead of by ascending subpool number as in MVS/370. To obtain storage by subpool ID, specify the new SUBPLST parameter.
SQA (changed)	x	x	x	No system trace entries are included because the trace buffers have been moved to the TRACE address space. SUM and TRT are the only dump options for including the system trace table in a SNAP/ABDUMP dump. SUMDUMP and TRT are the only options for including it in a system dump. The SQA option requests both the SQA area above and below 16 Mb.
SUBPLST (new)	x	x		Specified subpools, which appear in ascending order, as does the storage contained in each subpool. Another parameter for requesting subpool storage already exists, SPLS. However, SPLS causes all user subpools to be dumped. Also, the storage is printed in ascending address order instead of by subpool number.
SUBTASKS (new)	x			Program data (PDATA) information for subtasks.
SUM (new)	x		x	Summary dumps for abending tasks. See "User Summary Dumps."
TRT (changed)	x	x	x	The trace data included in dumps is changed. See "Changes to the System Trace Facility." The trace output is formatted the same in SNAP dumps, stand-alone dumps, and dumps printed via print dump. It is different from MVS/370 trace output. As in MVS/370, TRT causes trace data from the active trace facilities to be dumped. In MVS/XA, the dump can include master trace data, system trace data, and GTF data. In MVS/370, system trace and GTF cannot be active at the same time. Therefore, MVS/370 dumps never include both system and GTF trace data.

Figure 6-1 (Part 2 of 2). New, Updated, or Deleted Dump Options

New Symptom Dumps for Task-Mode Abends

When an ABEND, or program check, occurs and the ABDUMP module gets control, the system automatically issues a ten-line symptom dump. The symptom dump, which appears in the job listing, includes:

- The ABEND code and error ID
- The PSW at the time of the error, the instruction length code, and the interrupt code

- The name and address of the active load module, if the PSW points to an active load module
- The offset into the module where the error occurred, if the PSW points to an active load module
- Six bytes of storage before and six bytes after the PSW address at the time of the error
- The register contents at the time of the error

Unless the NOSYM parameter is specified in the appropriate PARMLIB member (IEAABD00, IEADMP00, or IEADMR00), all users get the symptom dump regardless of whether or not they supply a dump DD statement. However, TSO users must specify the WTPMSG parameter on the PROFILE command to see the symptom dump output. Installations that do not want symptom dumps can suppress them by specifying the NOSYM option in the appropriate PARMLIB member or by using the CHNGDUMP command. Specifying NOSYM in the IEADMP00 PARMLIB member also suppresses symptom dumps for users who do not specify dump DD statements.

The symptom dump is designed to help users identify duplicate problems and, in some cases, solve the problems without a full dump. Thus, symptom dumps can reduce the time required for problem determination. Note that symptom dumps are only issued for task-mode ABENDs.

User Summary Dumps

Users can now obtain a summary dump for abending tasks. A new SDATA keyword, SUM, requests a summary dump. Summary dumps show information that can help identify duplicate problems, followed by control blocks and storage areas. In many cases, the summary dump is sufficient to debug user program checks and ABEND dumps.

If SUM is the only dump option specified, the summary data for SYSUDUMP and SYSABEND dumps has the following format:

1. The dump title.
2. The ABEND code and PSW at the time of error.
3. The name and address of the load module in error.
4. The offset into the load module where the error occurred.
5. Control blocks (the same as if the CB option were specified).
6. Error control blocks (RTM2WAs and SCBs).
7. Save areas.
8. The general purpose registers at the time of error (from the RTM2WA).

9. The contents of the active load module and the load module associated with the last PRB.

10. One K of storage before and after the addresses pointed to by the PSW and registers at the time of error. The summary dump includes only storage areas for which the caller is authorized.

11. System trace table entries for the dumped ASID. (GTF records are not dumped.)

If SUM is not the only SDATA option, the summary data might be dispersed throughout the dump, depending on the other options specified.

The summary output produced for SYSMDUMP dumps is different from the summary output produced for other user dumps. You must use PRDMP and specify the SUMDUMP verb to get summary output. The output contains the same information and is in the same format as the summary information in SVC dumps.

SUM is a default option in the IEAABD00, IEADMP00, and IEADMR00 PARMLIB members. Unless installations delete the SUM options from those PARMLIB members or suppress them using the CHNGDUMP command, MVS/XA automatically produces a summary dump. In MVS/XA, SUM is the only default option in IEADMP00. Therefore, unless users that specify SYSUDUMP DD statements request additional data, they receive only summary data.

Dump Format Changes

The dump headers in user, SVC, and SYSMDUMP dumps contain additional information to aid in problem determination. Information in user dump indexes is presented differently. The SYSMDUMP and SVC printed summary dumps are restructured.

Changes to User Dump Headers

In MVS/XA, the dump headers in SNAP dumps caused by errors, SYSUDUMP, and SYSABEND dumps contain:

- The name of the load module that was executing at the time of the error.

- The offset into the load module, indicated by the PSW. The offset points the user to the failing instruction or to the next sequential instruction at the time of the error.

The SYSMDUMP dump header contains a new symptom buffer to help users identify duplicate problems or solve problems without a full dump. "New Symptom Dumps for Task-Mode Abends" describes the information contained in the symptom buffer.

User Dump Indexes

The indexes of SYSUDUMP, SYSABEND, and SNAP dumps list alphabetically the name of each active load module and the page number in the dump where it starts.

Changes to SYSMDUMP and SVC Dump Formats

The printed summary dump is restructured to make it easier to use. Following is a summary of the changes:

- Individual control blocks are formatted and dumped as separate logical records with unique IDs.
- The general, unformatted storage areas are printed in ascending address order within an address space.
- The dump index gives the starting page number of all formatted storage areas. It also has entries for each storage area requested.
- Trace data is no longer included in the formatted summary output. The trace table appears in the main body of the dump and can be formatted using IPCS or the TRACE verb of PRDMP.

The dump header records of SVC dumps and SYSMDUMP dumps contain the following additional information to aid in problem determination:

- The name of the active load module at the time of the error, if that information is available in the SDWA.
- The offset into the active load module of the instruction that caused the ABEND (in SYSMDUMP headers only).
- The PSW at the time of the error (in SYSMDUMP headers only).
- Six bytes of storage preceding the PSW address at the time of the error and six bytes following the address (in SYSMDUMP headers only). The failing instruction will be in one of those six-byte areas.
- The current SDWA control block, if available.
- Flags indicating whether SYSMDUMP or SVC dump produced the dump.
- The ID of the processor on which the dump was initiated.
- The name of the dump data set.

Note that the SYSMDUMP dump header record contains all of the information available in user symptom dumps, but in a different format.

Vector Registers in Dumps

If the Vector Facility is installed and functioning, the contents of the vector registers appear in SNAP, ABDUMP, and stand-alone dumps. For SNAP and ABDUMP dumps, the contents of the vector registers appear in the listing along with the contents of the other registers associated with the processor running the job.

When tasks using the Vector Facility are interrupted, the contents of the vector registers are saved in a control block known as the vector status save area (VSSA).

Formatting the Contents of Vector Registers

There are four ways to format the contents of the vector registers:

1. The PRDMP CPUDATA control statement formats the vector registers in dumps taken by stand-alone dump (SADMP).

2. The interactive problem control system (IPCS) STATUS subcommand formats the vector registers in dumps taken by stand-alone dump (SADMP).

3. The FORMAT keyword of the IPCS SUMMARY subcommand and the FORMAT parameter of the PRDMP SUMMARY control statement detect the presence of saved vector registers and invoke the vector formatter so that saved vector information is formatted with other TCB information.

4. The SNAP macro produces a dump listing that includes the formatted contents of the vector registers.

Suppressing Dumps

The SLIP command has new operands for selectively suppressing dumps. MVS/XA uses SLIP commands to suppress user and system dumps that are normally not required for problem determination. Release 2.1.1 introduces dump analysis and elimination (DAE), a new SDUMP function that suppresses duplicate dumps.

New Operands on the SLIP Command for Suppressing Dumps

The SLIP ACTION keyword has new operands, NOSYSA, NOSYSU, NOSYSM, and NOSVCD, that separately suppress SYSABEND, SYSUDUMP, SYSMDUMP, and SVC dumps, respectively. The new operands make the command more versatile. Installations can suppress specific types of dumps for ABENDs without suppressing all types.

MVS/XA's Use of SLIP Commands

MVS/XA uses SLIP commands to automatically suppress user and system dumps for ABEND codes that normally do not require a dump for problem determination. Examples are ABEND codes x'B37', x'D37', x'E37' and x'80A' (out-of-space ABENDs). The SLIP commands that suppress those dumps are in a new PARMLIB member, IEACMD00.

When a system dump is suppressed, the system puts the SVC dump status code in the appropriate LOGREC entry, as it does in MVS/370. When a user dump is suppressed because of a SLIP command, the system issues a write-to-programmer (WTP) message to inform the user.

If your installation already uses SLIP commands to suppress dumps, compare your list with the SLIP commands in IEACMD00. Delete all unnecessary commands to conserve SQA space. SLIP commands in your PARMLIB member (COMMNDxx) override commands in IEACMD00. Therefore, if any commands in IEACMD00 are undesirable, delete them or add SLIP commands to override them in COMMNDxx. Keep in mind that IEACMD00 might be refreshed with subsequent system updates.

When the system processes IEACMD00 at IPL time, it allocates fixed storage for the SLIP action processors and the control blocks they use. "Fixed Storage for SLIP Command Processors (IEASLPxx)" describes the fixed storage requirements.

Dump Analysis and Elimination (DAE)

Dump analysis and elimination (DAE) is a new SDUMP function in Release 2.1.1 that collects symptom data before taking SYSMDUMP or SVC dumps. DAE uses that data to identify and suppress duplicate dumps. If DAE does not suppress a dump, you see the symptom data in the dump header record.

By means of the SET DAE=xx operator command and appropriate code in recovery routines, you can request that DAE do one or more of the following for each dump type (SYSMDUMP and SVC):

- **MATCH** - Determine if the symptom data matches data already collected for the same dump type. Depending on other criteria described later, the system either suppresses duplicate dumps or reports matches in dump header records.

- **UPDATE** - Record the symptom data in the SYS1.DAE data set. Using UPDATE implies MATCH. If the data already appears in SYS1.DAE (that is, a problem with the same symptoms has already occurred), instead of creating an identical entry, DAE adds one to the incidence count in the existing entry. Incidence counts thus indicate the number of times particular problems have occurred. They appear in dump header records along with the symptom data.

  If you do not request updating, the system keeps the symptom data in storage elsewhere, but only until DAE processing is stopped. Consequently, DAE cannot use that data for comparison the next time it is active.

- **SUPPRESS** - Suppress dumps having symptom data that matches data already collected for the same dump type. Using SUPPRESS also implies MATCH. Instead of duplicate SYSMDUMPs, users receive message IEA838I, which contains the symptom data. DAE does not issue a message for suppression of SVC dumps. If you request dump suppression, the dump header records of dumps that are NOT suppressed indicate why.

  Note: DAE will not suppress a duplicate dump, even when SUPPRESS is requested, when one of the following occurs:

  - When you use the following parameters on the SLIP command to prevent DAE from suppressing dumps: SVCD (SVC dump), TRDUMP (trace dump), and NOSUP (no suppression). NOSUP is new in Release 2.1.1. See the SLIP entry in Figure 4-3 for more detail.

  - When the recovery routine that calls RTM to take the dump fails to:

    1. provide the symptom information that DAE requires in the ABDUMP symptom area, SDWA, SDWAVRA, or SDWA extensions

    2. supply the VRADAE key (via the VRADATA macro) in the SDWAVRA.

  As of the releases indicated below, the following components' recovery routines allow DAE to suppress dumps they produce:

Releases	Components Using DAE
Release 2.1.1	DAE, allocation, converter/interpreter, display dump command processor, DUMPDS command processor, and scheduler JCL facility.
Release 2.1.2	contents supervision, global resource serialization, SRM, TRACE, and VSM.
Release 2.1.3AE	availability manager

Controlling DAE Processing

You control DAE processing via records in the new ADYSETxx PARMLIB members. Each record can specify:

- Whether DAE is to be started or stopped
- The type of processing DAE is to perform for each dump type (UPDATE, MATCH, SUPPRESS, or a combination)
- How many dump entries DAE can store in the SYS1.DAE data set

IBM supplies three ADYSETxx members in Release 2.1.1: ADYSET00, ADYSET01, and ADYSET02. ADYSET00 and ADYSET02 are the same. Both:

- Start DAE processing.
- Request UPDATE, MATCH, and SUPPRESS processing for SYSMDUMPs and UPDATE and MATCH processing for SVC dumps.
- Allow DAE to store up to 400 entries in SYS1.DAE.

ADYSET01 stops DAE processing. If you require different options, create additional ADYSETxx members or modify ADYSET02.

A new SET DAE=xx command specifies which ADYSETxx member MVS/XA is to use. Only one member can be in effect at a time. However, you can issue a SET DAE command at any time to change DAE processing.

The IEACMD00 PARMLIB member shipped with Release 2.1.1 includes the command SET DAE=00. Thus, unless you change the defaults, during initialization, the system automatically starts DAE processing with the options specified in ADYSET00.

You also have some control over:

- The symptoms DAE collects for each dump type
- The minimum number of symptoms that must match before DAE considers the dump a duplicate
- The minimum number of bytes of meaningful data each symptom must contain before DAE can use it

A new non-executable load module, ADYDFLT, contains default symptoms and specifies which symptoms are required and which are optional. It also establishes the minimum number of bytes the symptoms must contain. *SPL: System Modifications* describes how to add to those defaults using the VRADATA macro.

Actions to be Taken

Before performing an IPL:

- Create a SYS1.DAE data set. If SYS1.DAE is not cataloged at IPL time, the operator receives a message stating that attempts to start DAE processing failed.: *SPL: System Modifications* describes how to create SYS1.DAE using JCL in the DAEALLOC member of SYS1.SAMPLIB. Consider allocating SYS1.DAE with DISP=SHR so you can browse, add, or delete records in the data set. You might, for example, want to delete information related to a problem after applying service to fix it. Note also that you cannot share SYS1.DAE among systems.
- Ensure that an ADYSETxx member specifies the desired DAE options.

- Either ensure that the IEACMDxx member MVS/XA uses contains the appropriate SET DAE=xx command, or instruct the operator to issue that command.

New and Updated PRDMP Control Statements

The following figure describes the print dump (PRDMP) control statements that Release 2.1.x adds, updates, or deletes. For more information, see *SPL: Service Aids*.

Statement	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
ASMDATA	x							Requests auxiliary storage management (ASM) control blocks. ASMDATA no longer formats RSM control blocks, as it did in MVS/370.
AVMDATA						x		Formats and prints availability manager control blocks.
DAEDATA		x						Formats and prints the error data (symptoms) that dump analysis and elimination (DAE) collects. A new DAE header exit, ADYHDFMT, puts the same information on the title page. DAEDATA causes PRDMP to repeat the DAE symptoms later in the dump. It is designed mainly for use at the terminal.
DISPLAY	x							DISPLAY is deleted in MVS/XA. Interactive Problem Control System (IPCS) provides equivalent interactive dump scanning functions. See the IPCS information in this chapter for more information.
FORMAT	x							Has new operands to limit formatting and printing to selected address spaces. The new operands are ASID, JOBNAME, CURRENT, ERROR, and ALL. The CURRENT address spaces are the primary, secondary, home, and CML lock holder's address spaces. When FORMAT is specified without operands, PRDMP formats and prints control blocks from the CURRENT and ERROR address spaces only. In MVS/370, the FORMAT statement has no operands. PRDMP formats and prints control blocks from all address spaces contained in the dump. To get that same output in MVS/XA, you must specify the ALL operand.
			x					Beginning with Release 2.1.2 the FORMAT control statement works as though it were a SUMMARY FORMAT statement. It formats major system control blocks and data associated with each address space in the dumped system.
IOSDATA	x							Requests IOS control blocks.
JES2			x					Formats the contents of specific control blocks within the JES2 address space.
JES3			x					Formats the contents of specific control blocks within the JES3 address space.
LPAMAP	x							Requests the names of all modules in the link pack area of the dumped system or on the LPA active queue at the time of the dump. In MVS/370, PRDMP lists only the names of modules on the LPA active queue at the time of the dump. LPAMAP also has new operands, EPA and MODNAME, to indicate how lists are to be ordered. The EPA operand requests sorting by entry point address. MODNAME requests alphabetical sorting by module name. If neither is specified, the printed output includes lists sorted both ways.
MTRACE		x						Formats and prints the master trace table for the dumped system. The console messages appear in the order in which they were issued.
NUCMAP	x							Requests the names of system modules in the nucleus when the dump was taken. The list includes the name, entry point, entry point attributes, and length of each module. Note that the NUCMAP statement does not produce a map of storage in the nucleus. To get a map of storage, use either the PRINT STORAGE= control statement or the AMBLIST utility.

Figure 6-2 (Part 1 of 2). New, Updated, or Deleted Print Dump Control Statements

Statement	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
RSMDATA	x							Requests real storage management (RSM) control blocks.
SADMPMSG				x				Formats and prints all the messages written by the SADMP virtual storage dump program including some that do not appear on the operator console. These messages appear in the message log even if you specified MSG=ACTION when you generated the SADMP program. The message log does not contain any of the messages produced by the SADMP real storage dump program.
SUMMARY			x					Release 2.1.2 adds keywords that specify: (1) for which address spaces the system is to collect information, and (2) the information to be collected. The new keywords are the same as the keywords on the IPCS SUMMARY subcommand, which is also changed. See "New and Changed IPCS Subcommands."
TCAMMAP			x					Formats and prints the contents of selected ACF/TCAM control blocks included in dumps produced by the SADMP and SVC dump programs.
TRACE	x							Requests trace data. Programmers must specify the new TRACE verb to obtain the system trace table in PRDMP output. TRACE also allows users to limit the trace data printed. See *SPL: Service Aids* for a description of trace operands.
VSMDATA	x							A new verb that requests virtual storage management (VSM) control blocks.
			x					Release 2.1.2 provides keywords for limiting the output. The following keywords request the VSM control blocks in the private areas of specific address spaces: ASIDLIST - Address spaces whose ASIDs are listed on the ASIDLIST keyword. JOBLIST - Jobs listed on the JOBLIST keyword. You can use JOBNAME as an alias for JOBLIST. CURRENT - Address spaces that were active when the dump was taken. ERROR - Address spaces that terminated abnormally (ACSBCC≠0) or that contain TCBs representing tasks that completed abnormally (TCBCMP≠0 or TCBRTWA≠0). TCBERROR - Address spaces that include TCBs representing tasks that terminated abnormally (TCBCMP≠0 or TCBRTWA≠0). TCBERROR specifies a subset of the information requested using the ERROR keyword. ALL - All address spaces in the system. NOASIDS - Prevents the system from including VSM control blocks from the private areas of any address spaces. You can also limit output using: GLOBAL - Requests the VSM control blocks in the SQA and CSA. NOGLOBAL - Prevents the system from including VSM control blocks in the SQA and CSA. In previous releases, VSMDATA has no keywords. It requests VSM control blocks from the private areas of all address spaces, the SQA, and the CSA. To get that same information in Release 2.1.2, you must specify the ALL and GLOBAL keywords. If VSMDATA is specified alone, you get only the VSM control blocks that the ERROR, CURRENT, and GLOBAL keywords request.

Figure 6-2 (Part 2 of 2). New, Updated, or Deleted Print Dump Control Statements

Print Dump Index

The MVS/XA version of PRDMP writes a dump index and allows user exits to insert their own entries into the index. See *SPL: Service Aids.*

The MVS/XA PRDMP procedure in SYS1.PROCLIB includes a new DD statement requesting that PRDMP write the dump index to a sequential data set other than the PRINTER data set. Using a separate index data set allows you to print the index before the dump.

To have the index printed at the beginning of the dump, either use the PRDMP procedure in the MVS/XA PROCLIB, or add a DD statement in your own PRDMP procedure in PROCLIB. For example:

```
//INDEX  DD  SYSOUT=A
```

To obtain the index in front of the dump, the INDEX DD statement must precede the PRINTER DD statement. Unless the INDEX DD statement is specified, PRDMP prints the index on the PRINTER data set after the dump.

Print Dump Requirements for Printers

Print dump (PRDMP) output lines are 132 characters long. If your installation uses a printer with a line length of less than 132 characters, you might lose information.

New and Changed IPCS Subcommands

The IPCS subcommands listed in the following figure were added or changed during Release 2.1.x. See the *MVS/XA Conversion Notebook, Volume 2* for a similar chart of Release 2.2.x changes. Only the changed EVALDUMP subcommand is incompatible with previous releases of IPCS.

Subcommand	Description of Change
EVALDEF	The variable list on the CLIST and DIALOG keywords can include SOURCE(variable name). The new term is an alternative to using DATASET(variable name) and DSNAME(variable name).
EVALDUMP	You must use one of the following keywords to specify the dump source: ACTIVE|MAIN|STORAGE DSNAME(dsname)|DATASET(dsname) FILE(ddname)|DDNAME(ddname) Previous IPCS releases expect the second positional parameter to be the data set name. Therefore, you need to change programs that use the EVALDUMP subcommand. The formatted output from EVALDUMP now includes the keyword ACTIVE, DSNAME, or FILE before the dump source. Previous IPCS releases formatted the data set name without the keyword and surrounding parentheses. You might have to modify programs because of this change as well. The variable list on the CLIST and DIALOG keywords can include SOURCE(variable name). The new term is an alternative to using DATASET(variable name) and DSNAME(variable name).
LISTDUMP	The format of the output is changed to display the dump source.
OPEN	You can now designate on the TITLE keyword the timestamp IPCS is to include on the top of every print file page. The default timestamp is the time problem analysis started. You might want change it to the time the dump was taken.
RENUM	A new subcommand that renumbers the symbols in the symbol stack so that every numeric suffix between the first and the last symbol is used.
RUNCHAIN	A new EXEC keyword specifies a CLIST statement or an IPCS subcommand that IPCS executes each time it processes a control block in the chain. That is, IPCS locates a control block, processes it as requested, then executes the statement or subcommand on the EXEC keyword before continuing to the next control block. Thus, you can use the EXEC keyword on RUNCHAIN for iterative processing that previously required several statements. You can, for example, use RUNCHAIN to look at the RBs queued from a chain of TCBs. On the RUNCHAIN command that searches the TCB chain, use an EXEC keyword that specifies another RUNCHAIN command, one that searches the RB chain. The system then displays the first TCB, all of the RBs chained to it, the second TCB and its RBs, and so on.
SETDEF	The format of the output is changed to display the dump source.
STACK	A new subcommand that creates a symbol in the symbol stack. It is similar to EQUATE, except that you need not specify the symbol to be created. IPCS always creates the symbol using the next suffix after the largest one used.
STATUS	As of Release 2.1.3VFE, the STATUS subcommand formats the vector registers taken by stand-alone dump (SADMP).
Subcommands that specify the dump source	Except for ADDDSN, DELDSN, LISTDSN, and MODDSN, all IPCS subcommands that accept dump data set names are changed to accept these additional dump source keywords: ACTIVE|MAIN|STORAGE FILE(ddname)|DDNAME(ddname) The existing keywords, DSNAME(dsname) and DATASET(dsname), are still valid. For more information, see "Accessing Additional Sources of Dump Data Using IPCS."

Figure 6-3 (Part 1 of 2). New and Changed IPCS Subcommands

Subcommand	Description of Change
SUMMARY	Several new keywords provide additional options that specify: (1) for which address spaces the system is to collect information, and (2) the information to be extracted. Also, some old keywords have been replaced by similar ones that request the same information. These changes are to make the IPCS SUMMARY subcommand like the PRDMP FORMAT and SUMMARY FORMAT verbs, which are also changed. Following are the keyword changes: ALL - Requests information from all address spaces in the system. In previous IPCS releases, if you do not specify from which address spaces IPCS is to take information, the default is all address spaces. In this release, the default is the ERROR and CURRENT address spaces. ASIDLIST - Replaces the ASID keyword. Like ASID, it specifies a list of ASIDs to be processed. The list can include a range of ASIDs, which is not allowed in earlier releases. The system accepts ASID as an abbreviation of ASIDLIST, so the change is compatible. JOBLIST - Replaces the JOB keyword. Like JOB, it specifies a list of job names whose associated address spaces are to be processed. To distinguish JOBLIST from the new JOBSUMMARY keyword, the minimum abbreviation for JOBLIST is JOBL. JOBNAME is also an acceptable substitute. You need to change any programs that specify JOB or J to request JOBLIST information. CURRENT - Requests information from all address spaces that were active when the dump was taken. ERROR - Requests information from address spaces that terminated abnormally (ASCBCC≠0) or that contain TCBs representing tasks that completed abnormally (TCBCMP≠0 or TCBRTWA≠0.) TCBERROR - Requests information from all address spaces that contain TCBs representing tasks that completed abnormally (TCBCMP≠0 or TCBRTWA≠0). TCBERROR specifies a subset of the address spaces that ERROR selects. ANOMALY - Requests different information in Release 2.1.2 than in previous releases of IPCS. In Release 2.1.2, ANOMALY requests the same information as TCBERROR. In previous IPCS releases, it requests a subset of the TCBSUMMARY information. FORMAT - A new keyword that requests major system control blocks and data associated with each address space in the dumped system. FORMAT produces the same output as the PRDMP FORMAT and SUMMARY FORMAT verbs. KEYFIELD - Requests key fields in the ASCBs, TCBs, and RBs of the specified address spaces. The output matches the output from the PRDMP SUMMARY verb. If you do not request specific information on the SUMMARY subcommand, IPCS gives you the information that KEYFIELD requests. JOBSUMMARY - Requests the following summary information: - A list of active CPUs - Scheduled services - For each address space specified: the jobname, ACSB location, ASID, status of the address space, local service manager queue, local service priority queue, TCB locations, completion codes, and whether or not the TCBs were active at the time of the dump TCBSUMMARY - Produces the same output, but in a different format.

Figure 6-3 (Part 2 of 2). New and Changed IPCS Subcommands

Accessing Additional Sources of Dump Data Using IPCS

The IPCS component of Release 2.1.2 can access: (1) data sets stored on tape as well as direct access devices, and (2) main storage of the MVS/XA address space in which IPCS is executing. Also, multi-volume data sets no longer need to contain fixed-length records. Earlier releases of IPCS can access only cataloged data sets on direct access devices. Multi-volume data sets previously had to contain fixed-length records.

To access the additional sources of dump data, you can now specify the keywords listed below on most IPCS subcommands that specify the dump source. The only subcommands that do not accept the keywords are ADDDSN, DELDSN, LISTDSN, and MODDSN:

```
ACTIVE, MAIN, or STORAGE

FILE(ddname) or DDNAME(ddname)
```

ACTIVE, MAIN and STORAGE all request that information be taken from main storage. FILE and DDNAME both specify ddnames currently associated with a dump data set. Three subcommands, CLOSE, DROPDUMP, and OPEN, allow more than one ddname in parenthesis.

Four other subcommands, LISTDUMP, SETDEF, EVALDEF, and EVALDUMP, are also changed to support new dump sources. Only the change to EVALDUMP is incompatible. See "New and Changed IPCS Subcommands" for descriptions of the changes.

Support for additional dump sources calls for new rules regarding which dump data sets you can move without invalidating the data set's dump directory entry. You can safely move:

- Data sets containing fixed length records to direct access devices
- Any data set to a single reel of tape

New IPCS Panels

Release 2.1.2 includes two new IPCS panels, BLSPDISE and BLSPDSLE. BLSPDISE is a top selection panel that, when hooked into the ISPF primary option menu, provides a convenient way of initiating IPCS dialogs. The two selections on BLSPDISE are:

- BROWSE, which invokes BLSLDISP, the full-screen dump viewing dialog program.

- IPCS, which displays the other new panel, BLSPDSLE. BLSDPSLE allows you to enter IPCS subcommands.

Your installation might have created similar panels in the past using instructions in the *IPCS User's Guide and Reference*. BLSPDISE is similar to the IPSELCT panel described in the guide. BLSPDSLE is identical to IPCMD, which is also described.

Changes to the IPCS BROWSE Panels

Following are several changes in the way you use the Release 2.1.2 BLSLDISP panels. Most of the changes are to make BLSLDISP more like the ISPF BROWSE and EDIT panels.

- To identify on entry panels the storage IPCS is to display, use one of the following keywords:

  ```
  ACTIVE, MAIN, or STORAGE
  DSNAME(dsname) or DATASET(dsname)
  FILE(ddname) or DDNAME(ddname)
  ```

  Because IPCS can now access main storage or data sets using a ddname, you can no longer simply specify a data set name.

- Instead of displaying on storage panels repetitive data or blanks for storage that cannot be obtained, IPCS inserts a one-line summary, either:

```
LENGTH(xxxxx)==>  Storage not available
LENGTH(xxxxx)==>  All bytes contain X'xx' (or C'c')
LENGTH(xxxxx)==>  Same as above
```

- On pointer and storage panels, you can now use the following primary commands:

  STACK — Adds a symbol to the symbol stack.

  RENUM — Renumbers the symbol stack so that every numeric suffix between the first and last symbol is used.

  FIND — Locates and displays storage containing specified data.

  RFIND — Repeats the last FIND command. RFIND is disabled in Release 2.1.2. You can enable it by creating a command table for IPCS, as described in the *IPCS User's Guide and Reference*.

- You can use two new operands, CURSOR and X, on the primary commands you enter on storage panels. CURSOR represents the fullword that the cursor precedes or is under. IPCS treats that fullword as the target address of a command. For example, LOCATE CURSOR% displays the storage beginning at the 24-bit address in the byte the cursor precedes or is under when the command is executed.

  X represents the first byte of the displayed storage. IPCS treats the contents of that byte as the target address of the command.

- You can put address space keywords on STACK and LOCATE subcommands. Thus, you can display data from an address space other than the one currently displayed without leaving the storage panel.

- IPCS displays the output from IPCS subcommands and dump processing exits in full-screen mode rather than line mode.

- You can update the dump's symbol stack from more than one logical screen. Previously, if working in split-screen mode, you could update the stack from only one screen.

  For more information, see either the *IPCS User's Guide and Reference* or the tutorial panels for the BLSLDISP dialog program. You can access the online tutorial by entering HELP on the command line of any BLSLDISP panel.

Changes to the Titles of IPCS Print Files

Beginning with Release 2.1.2 IPCS print files have different default titles. Also, instead of DATE and TIME headings, the first line of each page contains a 17-character timestamp.

The default title is the title in the default dump data set. If no title is available, IPCS uses the old default, "IPCS PRINT LOG FOR userid." As in previous releases, you can override the default by specifying a different title on the OPEN subcommand that opens the print file.

Although the format of the date and time is changed, the default values are still the date and time problem analysis started. As of Release 2.1.2, however, you can specify a different value (for example, the time the dump was taken) on the TITLE keyword of the OPEN subcommand.

Using the MVS/XA Versions of IPCS and PRDMP on Other Systems

To aid in migrating to MVS/XA, IBM allows you to execute MVS/XA versions of IPCS and PRDMP on certain MVS/SP Version 1 systems. You need the MVS/XA versions to view and print MVS/XA dumps. The MVS/370 versions of IPCS and PRDMP can process only MVS/370 dumps; the MVS/XA versions can process only MVS/XA dumps. In fact, PRDMP erases dumps taken on different versions of MVS.

IBM imposes some restrictions on running the MVS/XA modules on MVS/370 systems. The MVS/370 processor must be in a location where MVS/SP Version 2 and MVS/XA DFP are licensed. You can use IPCS and PRDMP on the MVS/370 system up to 18 months after the first shipment of MVS/XA program products to that location. The Agreement for IBM Licensed Programs (Z120-2800) defines the term "location."

The remainder of this topic describes how to obtain the modules and data sets required to run MVS/XA PRDMP and IPCS on an MVS/370 system. You might also want to run the PRDMP and IPCS programs from one release of MVS/XA on an earlier release. Although you can use any MVS/XA release of PRDMP or IPCS to print or view dumps taken on another MVS/XA system, to format all control blocks correctly, the level of PRDMP or IPCS must match the level of the dump.

IBM provides the job streams required to copy the MVS/XA IPCS and PRDMP modules (except those supporting the JES2, JES3, VTAM, and EREP PRDMP exits) into a data set you can use on another system. In addition to creating that data set, you need to ensure that the IPCS modules access the correct IPCS/ISPF panel and message libraries. The way you perform these tasks depends on whether you are copying Release 2.1.2 or earlier levels of IPCS and PRDMP. This topic describes each method separately.

Finally, to obtain the EREP PRDMP exit, it is recommended that you have at least EREP Version 3 installed on your MVS/370 system. The EREP PRDMP exit which, as of EREP Version 2, is contained in EREP instead of in MVS/XA, is required to print MVS/XA LOGREC records. EREP Version 3 runs on both MVS/XA and MVS/370 and can process LOGREC records created on either.

Copying IPCS and PRDMP Modules and Data Sets

As of Release 2.1.2 the job streams for creating a data set containing PRDMP and IPCS modules are in several members of SYS1.ASAMPLIB and, after system generation, SYS1.SAMPLIB.

1. Combine the job streams into one member by running the job stream in the MIGJOB01 member. MIGJOB02 will then contain the combined job streams.

2. Replace the data set specification on the SYSLMOD DD statement with the name of your target data set. The default name on the SYSLMOD statement is SYS1.MIGLIB.

3. Edit the JOB statement in MIGJOB02 to reflect your account's requirements.

4. Run the job stream in MIGJOB02 to create the target data set.

   As of Release 2.1.2 MIGJOB02 needs a region size of 256 Kb.

As of Release 2.1.2 IPCS modules use panels and messages in two data sets, SYS1.SBLSPNL0 and SYS1.SBLSMSG0, respectively. Earlier systems with IPCS installed might also have data sets with the same names. Therefore, to ensure that IPCS uses copies of those data sets from Release 2.1.2 or subsequent releases:

1. Allocate two data sets in which to copy the panels and messages from Release 2.1.2 or subsequent releases. Give the data sets names other than SYS1.SBLSPNL0 or SYS1.SBLSMSG0.

2. Copy SYS1.SBLSPNL0 and SYS1.SBLSMSG0 data sets from Release 2.1.2 or later releases into the new data sets.

3. When allocating the data sets required to run IPCS from Release 2.1.2 or later releases, concatenate the new data sets in front of the ISPF message and panel data sets and, if included, the SYS1.SBLSPNL0 and SYS1.SBLSMSG0 data sets. (You can omit SYS1.SBLSPNL0 and SYS1.SBLSMSG0 from the concatenation.)

Note: As of Release 2.1.2 you must have ISPF Version 2 installed on the system in order to use the IPCS dialogs on your MVS/370 system..

Beginning with Release 2.1.2 the PRDMP procedure for starting PRDMP on either an MVS/XA or an MVS/370 system is different from the one for starting earlier PRDMP releases. As of Release 2.1.2, PRDMP runs as a command processor under TSO. Therefore, the EXEC and DD statements in the procedure are changed. See "SYS1.PROCLIB Changes" for a listing of the procedure to use beginning with Release 2.1.2.

Copying Release 2.1.0 and 2.1.1 IPCS and PRDMP Modules and Data Sets

The job streams for creating a data set that contains Release 2.1.0 or 2.1.1 PRDMP modules and compatible IPCS modules are in the PRDMPXA and BLSAMPLE members of SYS1.ASAMPLIB and, after system generation, SYS1.SAMPLIB:

1. Replace the data set specification on the SYSLMOD DD statement with the name of your target data set.

2. Edit the JOB statements in each member to reflect your account's requirements.

3. Run both job streams to create the target data set. (The PRDMP job stream copies component analysis routines that IPCS also uses into the target data set. Therefore, to use the MVS/XA level of IPCS on MVS/370, you must run both job streams.)

The Release 2.1.0 and 2.1.1 levels of IPCS use panels and messages contained in the SYS1.ABLSPNL0 and SYS1.ABLSMSG0 data sets, respectively. To guarantee access to the panels and messages:

1. Allocate two data sets in which to copy SYS1.ABLSPNL0 and SYS1.ABLSMSG0. Give the data sets names other than SYS1.SBLSPNL0 or SYS1.SBLSMSG0, because systems with IPCS installed might already have data sets with those names.

2. Copy SYS1.ABLSPNL0 and SYS1.ABLSMSG0 into the new data sets.

3. When allocating the data sets required to run MVS/XA IPCS, concatenate the new data sets in front of the ISPF message and panel data sets, and if included, the SYS1.SBLSPNL0 and SYS1.SBLSMSG0 data sets. (You can omit SYS1.SBLSPNL0 and SYS1.SBLSMSG0 from the concatenation.)

Debugging Considerations

Changes to the System Trace Facility

The MVS/XA system trace facility is significantly different from the MVS/370 version. The following list summarizes the differences:

- **Flexibility in selecting events to be traced**

  The MVS/XA system trace facility can perform explicit tracing, address space tracing, and branch tracing. Explicit tracing records all of the normal system interrupt and dispatch events traced in MVS/370, plus the following:

  Alternate CPU recovery interrupt (ACR)
  Lock suspension (SUSP)
  Machine check interrupt (MCH)
  New I/O instructions
  Restart interrupt (RST)
  Trace options alteration (ALTR)
  User-defined event trace (USRn)

  Address space tracing records successfully-executed PC, PT, and SSAR instructions. Branch tracing records successfully executed BALR, BASR, and BASSM instructions. (The system does not, however, trace branch instructions that do not branch out of line, for example, BALR x,0.)

  The TRACE command is changed to allow installations to dynamically control which type of tracing is performed. Options are:

  - Explicit and address space tracing on, branch tracing off
  - All tracing on
  - All tracing off

  The system treats explicit and address space tracing as a single option. Also, the system can perform branch tracing only when the other trace options are active.

- **System trace is automatically activated**

  The system automatically activates explicit and address space (but not branch) tracing at system initialization time. If you prefer other trace options, put an appropriate TRACE command in a COMMNDxx PARMLIB member or issue the command from the master operator console. In MVS/370, installations must use a TRACE command to keep system trace active after system initialization time.

- **Concurrent system and GTF tracing**

  System and GTF tracing can be active at the same time on an MVS/XA system. Activating GTF trace no longer turns off system trace, as it does in MVS/370.

  Explicit system tracing and GTF tracing record some of the same events. Therefore, if you activate both, you might want to tailor GTF trace to record only events that explicit system tracing does not record. *Diagnostic Techniques* lists the events that system trace records. *SPL: Service Aids* describes the events that GTF trace records and how to tailor GTF trace.

- **The structure, location, and format of the system trace table is changed**

  The system trace table consists of queues of trace buffers, one queue for each processor sharing the operating system. The system trace table formatter merges the entries from the separate trace buffers into a single logical table. In MVS/370, the system trace table is a single buffer.

  The trace buffers are located in the LSQA of a new TRACE address space. In MVS/370, the system trace table is located in SQA. Moving the trace data reduces the system's use of common virtual storage. It also isolates the trace data from the rest of the system, which provides a greater degree of data integrity.

  System trace entries vary in length. MVS/370 entries have fixed length.

- **Installations can control the size of the trace table**

  The TRACE command is changed to allow installations to dynamically change the size of the trace table. The default size is 16 K of trace buffers per online processor. The size of the MVS/370 trace table is fixed at IPL time.

- **Installations can create and format their own trace entries**

  Installations can use a new macro, PTRACE, to create their own trace table entries. *System Macros and Facilities* describes PTRACE. *Diagnostic Techniques* describes how to create and format user entries.

- **Dumping trace data**

  TRT, SUM, and SUMDUMP are the only dump options for including the system trace data in dumps. The SQA option no longer dumps system trace entries because the trace buffers have been moved to the TRACE address space. The system trace data printed in **user dumps** depends on the requestor's authorization. If the requestor is authorized, the dump includes

the system trace table entries for all address spaces. If the requestor is unauthorized, the dump includes only system trace entries from the current address space that were made after the job started. Dumping only job-related trace entries for unauthorized users improves system integrity and makes debugging problem programs easier.

SVC dumps include trace entries for all address spaces. The trace data always appears in the non-summary part of the dump, even when dumped in response to a SUM or SUMDUMP request.

Because the MVS/XA trace data is in separate buffers and the trace entries vary in length, it is not feasible to read unformatted dumps of the trace table. Installations need to use print dump (PRDMP) or SNAP/ABDUMP dump to format trace table entries. The system trace table formatter merges the entries from the separate trace buffers into a single logical table. The formatter merges timestamped entries (explicit trace events) from oldest to newest. It merges branch and address space trace entries, which are not timestamped, in relative order to the timestamped entries.

Note: To obtain formatted trace table entries you must include the new TRACE verb in the PRDMP procedure. The TRACE verb has operands that allow installations to limit the trace information printed. See "New and Updated PRDMP Control Statements."

SDWA Changes

The SDWA has increased in size. All of the additional storage is included in SDWA extensions. The additional storage contains data for I/O machine checks, new locks, new dump tailoring options that specify storage subpool lists, and new service data. The information is contained in the following extensions:

- The previously-existing recordable extension 1 (SDWARC1) contains additional service data.
- The new recordable extension (SDWARC2) contains I/O machine check data.
- The new recordable extension (SDWARC3) contains new lock and lockword information that can be specified in the FRELOCK keyword of the SETRP macro.
- A new non-recordable extension (SDWANRC2) contains the SNAP dump tailoring information for storage subpools.

Addressing Mode Reflected in Dumps

When producing summary (SUM or SUMDUMP) dumps, dump routines use the addressing mode at the time of the error to determine whether the addresses in registers are 24-bit or 31-bit values. If a program is running in 31-bit addressing mode when an error occurs, the system treats addresses as 31-bit values. If a program is running in 24-bit addressing mode, the system treats them as 24-bit values. If a program has 31-bit addresses in some registers and changes to 24-bit addressing mode just before an error occurs, the dump routines consider the addresses to be 24-bit values. As a result, dumps at times might include the contents of incorrect storage locations.

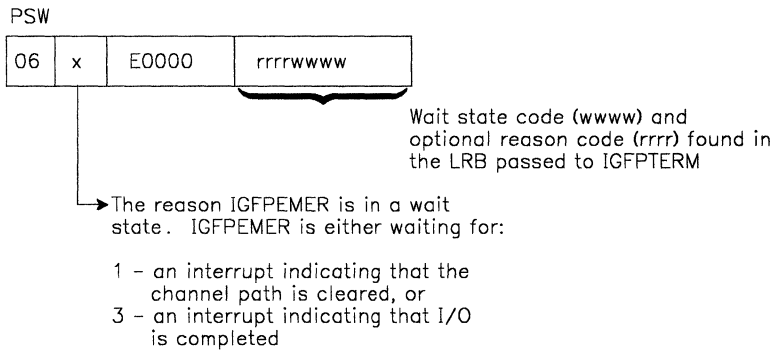
Specifying Reason Codes

Users can specify their own reason code on ABEND, CALLRTM, and SETRP macros. The user-supplied reason code supplements the completion code associated with abnormal termination. It allows users to uniquely identify the causes of abnormal termination that have meaning for their programs. RTM propagates the reason code to each recovery exit and to the TCB and ASCB control blocks so that the user-supplied reason code appears in system messages.

System Termination Facility Wait State Codes

In MVS/370, the system termination facility (IGFPTERM and IGFPTREC) issues wait state code X'024' when IGFPTREC fails to receive an I/O interrupt while attempting either to write a SYS1.LOGREC record or to issue a WTO message. Users are prevented from seeing the wait state code of interest, namely the wait state code indicating the error condition that caused system termination processing to begin (the wait state code in the LRB passed to IGFPTERM).

In MVS/XA, the system termination facility puts into the PSW the wait state code and the optional reason code found in the LRB, and a reason code indicating why IGFPEMER is in a wait state. (In MVS/XA, IGFPEMER replaces IGFPTREC.)



Programmers seeing a wait state code in a PSW with this format can locate the message that was to be displayed at the operator's console and the LOGREC record that was to be written to SYS1.LOGREC. At the time the wait state code is loaded, Register 1 points to a 2-word parameter list. The first word contains the address of the WTO message, the second word contains the address of the LOGREC record.

Exceeding the Region Limit

With MVS/XA, installations can use the SMF step initiation exit (IEFUSI) to specify region size and region limits. When IEFUSI changes the VSM region size and region limits, MVS/XA records the change in an SMF type 30 record. Therefore, if a job is cancelled for exceeding the limit when the JCL specified adequate space, check for an SMF record indicating that IEFUSI changed the limit. See "Limiting User Region Size using IEFUSI Instead of IEALIMIT" in Chapter 5 for a description of the IEFUSI enhancements.

If an internal error occurs in Release 2.1.2 and subsequent releases, checkpoint/restart puts diagnostic information into the SDWA for recording in SYS1.LOGREC. Checkpoint/restart also issues an SVC dump to store selected dump information in a SYS1.DUMPxx data set. The dump information includes:

- All storage currently allocated to checkpoint/restart
- 4 Kb of storage on each side of each register
- Load modules
- The SDWA

Dumps obtained using SYSUDUMP or SYSABEND DD statements are not useful for solving problems in checkpoint/restart. Use PRDMP or IPCS to print dumps that checkpoint/restart creates.

Chapter 7. Accounting

This chapter contains information pertaining to accounting procedures. In general, converting to MVS/XA does not require that you change your accounting programs significantly, if at all. You do, however, need to examine accounting programs to determine whether they will:

- Execute successfully in MVS/XA. Most SMF records are the same or compatibly expanded in MVS/XA. Therefore, in many cases, accounting programs will work unchanged.

- Bill jobs the same whether executed on MVS/XA or MVS/370 or on different releases of MVS/XA. After MVS/XA is installed and operating, you can perform comparison runs between your present system and MVS/XA. Depending on the results, you might need to adjust your accounting or billing algorithms.

  Although SMF reports additional measures of I/O activity and virtual storage use, it continues to report the old data as well. In most instances, the data is also derived the same way. Processor or CPU utilization times and EXCP counts of the number of physical blocks of data transferred for application data sets are calculated as in MVS/370. EXCP counts for program libraries, however, are slightly different. "Increases in EXCP Counts for Program Fetch Activity" describes the differences.

  With MVS/XA DFP Version 2 Release 1.0 a new address space, the catalog address space, is created. Catalog activity such as CPU and SRB time and EXCP counts are charged to this address space and not to the requesting address spaces as in previous releases. The accounting figures for a job may be affected depending on the amount of catalog activity required by the job. Catalog address space accounting information is recorded as an SMF type 30 subtype 6 record.

The topics in this chapter describe some changes that might affect existing accounting programs. Most of the information, however, describes new measurements you will want to use in the future, after your MVS/XA system is stabilized. The topics included are:

Device Connect Time

In addition to the EXCP counts available in MVS/370, SMF accumulates device connect time for each data set defined by a DD statement, for each address space, and for each command issued during a TSO session. Device connect time is similar to channel busy time in MVS/370. It measures the amount of time during an I/O operation that the channel subsystem is transferring data or control commands (such as SEEK) on the channel path. Device connect time is a more accurate measure of actual device use than the number of physical blocks transferred (the EXCP count).

Type 30 and 32 SMF records include new fields for reporting device connect time. In type 30 records, the SMF30DCT field in the EXCP section indicates the device connect time for a data set. The SMF30TCN field shows the total device connect time for the address space. The SMF32TCT field in type 32 records reports the total device connect time used while executing a command during a TSO session.

If you currently obtain EXCP counts from type 4, 5, 34, or 35 records and plan to use device connect time in MVS/XA, consider modifying those programs to obtain EXCP counts from type 30 and 32 records instead. Changing the programs now might ease the transition later. Device connect time is not reported in the other records mentioned.

New Fields Measuring Virtual Storage Use

The storage and paging section of type 30 SMF records includes new fields that report virtual storage use above and below 16 megabytes. Eventually, you might want to modify accounting routines that measure virtual storage to use the new data. Many system control blocks have moved to virtual storage above 16 megabytes. Also, user programs will begin using storage above 16 megabytes.

The new fields report:

- The region size below and above 16 megabytes (SMF30RGB and SMF30ERG)
- The maximum amount of virtual storage allocated from the LSQA and SWA subpools below and above 16 megabytes (SMF30ARB and SMF30EAR)
- The maximum amount of virtual storage allocated from the user subpools below and above 16 megabytes (SMF30URB and SMF30EUR)

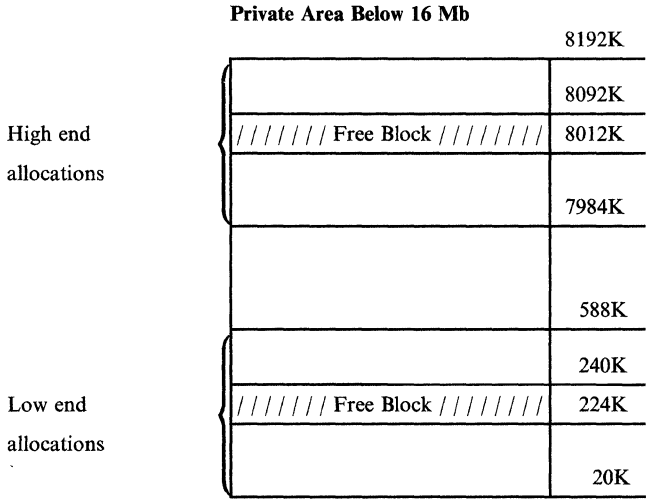
SMF30PRV and SMF30SYS Fields

The SMF30PRV and SMF30SYS fields continue to report private area use below 16 megabytes. However, MVS/XA calculates the source for the fields differently than MVS/370. In MVS/XA, SMF30SYS and SMF30PRV show the total number of bytes (in 1 K units) allocated from subpools in the high and low ends of the private area, respectively. The amounts do not include imbedded free blocks.

The MVS/370 values report the total number of bytes between the highest and lowest addresses allocated from subpools in the high and low ends, respectively.

The amounts include any free blocks imbedded in the respective ranges. Therefore, the MVS/XA values might be lower than the MVS/370 values for the same job.

The following picture illustrates the differences between the MVS/XA and MVS/370 values.



Field	*MVS/XA Value*	*MVS/370 Value*
SMF30SYS	128K	208K
SMF30PRV	552K	568K

Two new fields in the type 30 record (SMF30ARB and SMF30URB) report the same data as the MVS/XA SMF30SYS and SMF30PRV fields, but in bytes instead of 1 K units. SMF30ARB is equivalent to SMF30SYS. SMF30URB is equivalent to SMF30PRV.

Type 22 SMF Record Updates

MVS/XA replaces the channel section of type 22 records with channel path information. Therefore, you must at least reassemble existing programs that use type 22 records. You might also want to modify the programs to use the new channel path data.

Increases in EXCP Counts for Program Fetch Activity

The EXCP counts that SMF records for program fetch activity are likely to be higher in MVS/XA than in MVS/370. In both systems, SMF records EXCP counts in either the SMF30TEP field or, if a STEPLIB is used, in the SMF30BLK field or the equivalent SMF4EXCP field. (SMF30TEP and SMF30BLK are type 30 SMF records; SMF4EXCP is a type 4 record.) If your installation uses any of these fields to measure program fetch activity, you need to determine if the increase affects your accounting programs.

The higher counts result from program fetch changes described in "Ensuring Optimal Program Fetch Performance" in Chapter 8. MVS/XA records all fetch I/O activity, whereas MVS/370 misses some. (For example, it appears that MVS/370 does not count redrives caused by the need to fix additional storage.)

The Release 2.1.0 and 2.1.1 versions of program fetch count actual EXCPs. In systems with Release 2.1.2 program fetch or its equivalent (the version obtained by installing the PTF for APAR OZ75713 on Release 2.1.1), the EXCP counts for non-overlay modules with correct relocation dictionary (RLD) count values report the number of text blocks transferred instead of actual EXCPs. These counts are likely to be the same as the actual EXCP counts obtained in earlier MVS/XA releases. For overlay modules the SMF counts in Release 2.1.2 and its equivalent are likely to be less than in earlier MVS/XA releases, but more than in MVS/370.

Summary of SMF Record Updates

The following chart summarizes the SMF record updates for Release 2.1.x. Be sure to consult the corresponding chart in the *MVS/XA Conversion Notebook, Volume 2* for information on SMF record updates for Release 2.2.x. For more detail, see *SPL: SMF.*

SMF Record	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
Type 4 (Step Termination)			x					In Release 2.1.2, the SMF4RSHO field is increased from 2 bytes to 4 bytes to accommodate a region size greater than 16 megabytes. It is moved from the beginning to the end of the storage and paging section to avoid changing the offsets of other fields in the record. You must recompile programs that use the SMF4RSHO field.
Type 6 (JES2)			x					Includes several new fields at the end of the 3800 Printing Subsystem section. The new fields are meaningful only if the 3800 Printing Subsystem Model 3 is running under a functional subsystem (FSS).
Type 6 (JES2,JES3,PSF)					x			By installing Release 2.1.3VFE or by applying the PTFs for the APARs listed below to previous releases, installations have a restructured type 6 record. It has three extensions (3800, Routing, and 3800-3), any combination of which will be present depending on your level of MVS and JES2 or JES3. The header section of the restructured record includes bits that indicate when the 3800-3 extension is present (SMF6PAD1, bit 2 = 1) and when PSF has generated the type 6 record. (SMF6SBS, PSF = X'0007'). A record-level indicator field (SMF6INDC) has been created and is set to X'01' to indicate the type 6 record is a restructured record. In subsequent releases, each time a subsystem changes the type 6 record, it will also increment the record-level indicator. see *SPL: SMF* for details on the restructured type 6 record, and contact your IBM representative for information on the PTFs. For JES3 and PSF subsystems, the restructured type 6 record is incompatible with previous type 6 records. If you install the PTFs or Release 2.1.3VFE and use these subsystems, you must recompile all post-processing programs that use type 6 records. JES2 support for the type 6 restructure is incorporated in Release 2.1.5. Prior releases of JES2 use their own copy of the type 6 record and are not affected by the restructure. The only difference between the old and the new, restructured, JES2 type 6 record is the record-level indicator in the header. Therefore, the restructured SMF type 6 record can be used to map both the old and the new JES2 type 6 records. The APARs which apply to the restructure are: **SMF** OZ85601 or OZ85602 **JES3** OZ84504 **PSF** OZ85790 **SLR** PP41106

Figure 7-1 (Part 1 of 3). SMF Record Updates

SMF Record	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
Type 22 (Configuration Record)	x							The channel section of the configuration record is replaced by a channel path section. Also, the format of the storage section is changed to report 31-bit counters and addresses.
				x				A new extended storage section has been added to include the amount of extended storage on line at IPL time. Programs that use the mapping macro for this record only need to be recompiled if they need to access the extended storage data in the type 22 record.
					x			The CPU element section includes a new Vector Facility indicator flag to show which processors have a Vector Facility online. You need to recompile programs using the type 22 record only if you want to use the Vector Facility online status information.
							x	A type 22 record is written each time the ESTOR(E=id) keyword is used on the CONFIG command to configure extended storage online or offline. Field SMF22IND is set to 3 when the storage is taken offline and set to 2 when the storage is brought online.
Type 24 (JES2)					x			By installing Release 2.1.3VFE, or applying the PTFs for APAR OZ85828 to Releases 2.1.2 and 2.1.3, installations have this new record. It is used only by JES2 to record spool offload activity to offload data sets.
Type 30 (Common Address Space Work Record)	x							Includes additional fields described in "Device Connect Time" and "New Fields Measuring Virtual Storage Use." One other new field in the completion section, SMF30ARC, reports the abend reason code. Also, MVS/XA calculates the values in the SMF30PRV and SMF30SYS fields differently. See "SMF30PRV and SMF30SYS Fields."
			x					In Release 2.1.2, the SMF30RGN field is increased from 2 bytes to 4 bytes to accommodate a region size greater than 16 megabytes. It is moved from the beginning to the end of the storage and paging section to avoid changing the offsets of other fields in the record. You must recompile programs that use the SMF30RGN field.
					x			The processor accounting section includes the Vector Facility usage and Vector Facility affinity time. The processor accounting section also includes, for type 30 interval processing, the start time and date for the subtype 2 and 3 records. You need to recompile programs using the type 30 record only if you want to make use of the Vector Facility information or the subtype 2 or 3 interval start time and date.
Type 32 (TSO User Work Accounting Record)	x							Includes the device connect time per TSO user, in addition to the existing data.
Type 34 (TSO Step Termination)			x					In Release 2.1.2, the TIVEFRGN field is increased from 2 bytes to 4 bytes to accommodate a region size greater than 16 megabytes. It is moved from offset 74 to offset 82 to avoid changing the offsets of other fields in the record. You must recompile programs that use the TIVEFRGN field.
Types 4, 14, 15, 19, 30, 34, 40, 64, and 69	x							The adjacent fields containing channel addresses and unit addresses are combined to form a single field for a device number.

Figure 7-1 (Part 2 of 3). SMF Record Updates

SMF Record	Release							Description of Update
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7	
Type 4, 30, 34 and 40	x							VIO data sets are designated by the value X'7FFF' in the device address field. MVS/370 uses the value X'0FFF', which is a valid device number in MVS/XA.
Type 70-79	x							The formats of these records have changed. As a result: - Installations must use the post processor and report writers shipped with RMF Version 3 to process Version 3 SMF records. - Installations must modify user programs that process SMF records 70-79. *RMF Reference and User's Guide* describes the new format. Most fields contain the same information in Version 3 as they did in Version 2. - The RMF Version 3 post processor can process SMF records written by RMF Version 2 Release 2.2 (SE2 support) or later levels. The post processor converts the old records to the new format before processing them. Installations cannot assume that all data in the old records appears in the converted records. Because the contents of some records are changed (particularly those dealing with I/O operations), the post processor omits some data from the old records. The meanings of other fields have changed. - Installations that require data in SMF records written by an earlier level of RMF than RMF Version 2 Release 2.2 need to keep both the records and the earlier level of the post processor.
				x				In RMF Version 3 Release 3 that supports Release 2.1.3, SMF record types 71,78, and 79 have been updated incompatibly. Programs that use these records must be modified and recompiled: - Type 71 includes a new swap placement data section. The old swap count fields remain, but they are now filled with zeroes. - Types 78 and 79 include new subtypes (subtype 3 and subtype 14, respectively) for I/O queuing activity. - Type 79 also includes two new range fields for larger domain numbers.
					x			RMF provides three new fields in the type 70 CPU data section. Two support the Vector Facility and one is reserved for future use. SMF70VFS is useful for calculating vector affinity time. SMF70V is useful for ensuring that the affinity time in the post processor CPU duration report is valid. You need to recompile programs using the type 70 record only if you need to use the Vector Facility information. See *SPL: SMF* for more details on these changes to the type 70 record.
							x	RMF Version 3 Release 4.1 places a zero in field R791SEQN of the type 79 subtype 1 record.
Type 90				x				All of the fields in subtype 3 have increased from 1 to 2 bytes because Release 2.1.3 includes an increase in the maximum MPL from 256 to 999. Any programs using SMF type 90 subtype 3 records must be recompiled. Programs that need to handle subtype 3 records from both Release 2.1.3 and earlier releases can look at field SMF90RVN to determine which lengths and offsets to use. SMF90RVN contains X'02' if the subtype 3 record was produced by Release 2.1.3 and X'01' if it was produced by an earlier release.
All records	x							Bit 5 in the header is set to 1. The flag allows users to distinguish MVS/XA records from MVS/370 records.

Figure 7-1 (Part 3 of 3). SMF Record Updates

SMF Compatibility Between Release 2.1.0 and Later Releases

As of Release 2.1.1, to improve performance SMF changes data set handling in two ways:

- When formatting an SMF data set, SMF fills it with dummy records instead of binary zeros, as before. The dummy records are shorter than any valid SMF record and contain the characters 'SMFEOFMARK'. The SMF dump program, IFASMFDP, recognizes dummy records and terminates processing when it encounters one. Thus, IFASMFDP no longer reads to the physical end of file when processing partially filled data sets.

- SMF uses a binary search rather than a linear search to find where to start recording in a partially full data set. A binary search reduces the number of control intervals read before finding the starting point.

The Release 2.1.1 changes are compatible. Release 2.1.0 and later versions of IFASMFDP can read data sets with or without the SMF EOF marks. Although you can use the same SMF data sets for all levels of the system, you need to consider the following points:

- If you use the CLEAR or ALL options when running IFASMFDP, SMF formats the SMF data sets according to the release level of IFASMFDP.

- When processing data sets that contain SMF EOF marks, the Release 2.1.0 level of IFASMFDP ignores the marks and reads every control interval to the end of the data set. There is no performance gain. The Release 2.1.1 and later levels of IFASMFDP recognize the SMF EOF marks and terminate processing. You see the most significant performance gain when processing a large data set that contains SMF EOF marks and is almost empty.

- When processing data sets that do not contain SMF EOF marks, all levels of IFASMFDP read every control interval to the end of the data set. There is no performance gain.

- If you IPL a Release 2.1.0 system using SMF data sets that contain SMF EOF marks, the data sets appear full during normal data set selection. The operator must dump and clear at least one data set before SMF can begin recording.

- To find where to begin recording, the Release 2.1.1 level of SMF performs a binary search, regardless of the data set's format. The Release 2.1.0 level always performs a linear search. In neither case are any records lost.

- SMF initialization does not reformat any data set unless:

  - The data set has a bad control interval that caused an I/O error during the previous IPL. The data set was taken out of service after the error occurred, but the control interval is still bad.

  - The data set has just been allocated and has not been formatted.

In summary, you probably want to use the Release 2.1.0 level of IFASMFDP until your more current system is in production. You thereby avoid the situation where all SMF data sets appear full and, when running Release 2.1.1 or subsequent releases, you still have the advantage of a binary search.

Chapter 8. Measurement and Tuning

This chapter includes topics related to performance:

Ensuring Optimal Program Fetch Performance

Program fetch was rewritten in MVS/XA DFP Version 1 Release 1.0, modified in MVS/XA DFP Version 1 Release 1.2, and further modified in MVS/XA DFP Version 2 Release 1.0. These modifications improve program fetch performance and reduce the differences in handling overlay and non-overlay modules.

Installations with MVS/XA DFP Version 1 Release 1.1 obtain function equivalent to MVS/XA Version 1 Release 1.2 improvements by installing the PTFs for the following APARs:

OZ75713 - Replaces program fetch

OZ75717 - Changes the ALTERMOD and COPYMOD functions

OZ76136 - Changes a SYSGEN macro to ensure that future SYSGENs correctly include the new version of program fetch

Installations at the MVS/XA DFP Version 1 Release 1.2 level obtain functions equivalent to MVS/XA DFP Version 2 Release 1.0 by installing fixes for the following APARs:

OZ82525 - Replaces program fetch

OZ82530 - Changes IEBCOPY

OZ82528 - Linkage editor

All MVS/XA versions of program fetch can fetch the same modules as the MVS/370 version. However, for optimal performance, the MVS/XA DFP Version 2 Release 1.0 level of program fetch requires:

- A count value for each text block. The count value is the number of relocation dictionary (RLD), control, and RLD/control records associated with the text block.
- A count value of the number of text blocks in each overlay segment.
- Text blocks as large as the linkage editor allows for the output device.

"Recommended Actions" later in this topic describes how to modify program libraries to attain optimal fetch performance. The changes have no effect on the MVS/370 fetch process. The programs you can use to insert count values and reblock modules are:

- The linkage editors supplied with MVS/XA DFP[1], MVS/370 DFP, and DFDS 1.4[2].
- The ALTERMOD and COPYMOD functions of MVS/XA DFP and MVS/370 DFP versions of IEBCOPY. These functions of IEBCOPY in both MVS/370 DFP Release 2.1.1 and MVS/XA DFP Version 1 Release 1.1 can insert count values in and reblock only **non-overlay** modules (modules that are not in an overlay structure). These functions of IEBCOPY in MVS/XA DFP Version 2 Release 1.0 (and in MVS/XA DFP Version 1 Release 1.1 with the fix for APAR OZ75717 installed and MVS/XA DFP Version 1 Release 1.2) will insert count values in and can reblock **both overlay and non-overlay** modules.

---

1 The linkage editor supplied with MVS/XA DFP supplies the count of text blocks in an overlay segment. This function is included in MVS/XA DFP Version 2 and can be added to MVS/XA DFP Version 1 by means of APAR OZ82528 (PTF UZ78282).

2 You need to install the fix for APAR 0Z57635 on DFDS 1.4 to obtain correct counts. Modules link edited using the DFDS 1.4 linkage editor without the required PTF installed might contain incorrect counts. Incorrect counts have no effect on the MVS/370 fetch process. However, they degrade fetch performance in MVS/XA. If MVS/XA program fetch encounters an incorrect count value, it issues message CSV300I and continues without using count values.

The steps described in "Recommended Actions" correct any incorrect counts. If you take those actions, you need not separately search for and link edit modules with incorrect counts.

Performance Related Changes to the Linkage Editor and IEBCOPY

The linkage editor and IEBCOPY programs identified earlier record the number of relocation dictionary (RLD), control, and RLD/control records following each text block. They put the record count following the first text block in the load module's PDS directory entry. They record the counts for subsequent text blocks in the RLD/control or control record immediately preceding the text block. They also record the count of text blocks for each overlay segment in the corresponding note list entry.

Because the count values are located in existing fields that neither MVS/370 program fetch nor previous linkage editors use, load modules containing count values are downward compatible.

Performance Related Changes to Program Fetch

If valid counts are available, MVS/XA DFP Version 2 Release 1.0 program fetch reads one text record and up to 48 associated RLD, control, or RLD/control records using a single I/O operation. Program fetch uses program controlled interruptions (PCIs) to dynamically chain additional read operations to the channel program whenever possible. The PCI processing in MVS/XA DFP Version 2 Release 1.0 involves less disabled time than the PCI processing in MVS/370.

When the count values are invalid or missing, program fetch issues one I/O request for each text record and the first RLD or control record that follows, and one I/O request for each additional RLD, control, or RLD/control record. Therefore, fetch performance in MVS/XA depends on:

- Whether or not valid count values are present.
- The size of each text block. It is best to have block sizes as large as the linkage editor allows for the device type. The larger the block size, the more time program fetch has to chain additional read requests to the currently executing channel program. Chaining read requests improves performance by eliminating the need to:
  - Initiate separate I/O requests.
  - Perform SEEK operations if the access mechanism has been repositioned.
  - Re-establish the rotational position required to begin the read operation.

Recommended Actions

You can significantly improve fetch performance by inserting count values in modules that lack them, and by reblocking modules.

You can update modules using new operations that the MVS/XA DFP or MVS/370 DFP versions of IEBCOPY provide:

- ALTERMOD simply inserts count values.
- COPYMOD copies modules from one library to another. In the process, it inserts count values and reblocks the modules.

Using IEBCOPY with the COPYMOD parameter produces a new data set. Therefore, after copying the modules, you need to scratch the original data set and rename the new one.

The primary candidates for reblocking are:

- SYS1.LINKLIB
- SYS1.CMDLIB
- Program libraries used by interactive applications (for example, CICS and IMS, provided those programs use the standard program fetch)

Reblock the system libraries after constructing the system.

When using the IEBCOPY COPYMOD statement, you need to consider two parameters, MAXBLK and MINBLK, which specify the maximum and minimum block sizes IEBCOPY can create.

- Take the default MAXBLK value to obtain the largest block sizes the linkage editor supports for the device type.

- Use a MINBLK value of 1K. The initial default value for MINBLK is 1K; however, your installation might have changed it. *Utilities* describes how to reset MINBLK.

  Setting a small MINBLK default value might seem like a contradiction. However, the MINBLK value affects only the size of the last data record on a track. Because of the way program fetch chains read requests across tracks, that record can be small without negatively affecting program fetch performance.

You can also update modules by link editing them using any of the linkage editors identified on page 8-2. Unless you need to link edit a module for other reasons, however, using IEBCOPY is easier and faster.

Increasing the Size of the Page-fixed Area

Some MVS/370 installations improve fetch performance by increasing the amount of virtual storage program fetch fixes at one time. They make the change by adjusting a constant value within the page fix program. Because MVS/XA program fetch fixes 96K at one time, the equivalent modification is not required in MVS/XA. (MVS/370 program fetch fixes 18K. MVS/370 DFP program fetch fixes 64K. If you apply PTF UZ243577 to the MVS/370 program fetch, it fixes 18K for all input records up to 18K and an appropriate amount of storage for all input records greater than 18K and up to 32K.)

Maintaining Count Values and Optimal Block Sizes

To maintain count values and optimal block sizes when link editing the modules you modify, always use one of the linkage editors listed earlier. In addition, ensure that the linkage editor constructs the largest possible block size for the device being used. The linkage editor in MVS/XA DFP Version 2 allows maximum block sizes up to 32K. "Assembling and Link Editing Programs" on page 9-2 describes additional maintenance considerations. The following figure

summarizes how different versions of program fetch, IEBCOPY, and the linkage editor handle modules with and without count values.

INPUT	PROGRAM	OUTPUT/COMMENTS
A load module without count values	MVS/XA DFP linkage editor MVS/370 DFP linkage editor DFDS 1.4 linkage editor with the required PTF installed	A load module with count values inserted. Depending on the JCL used and other linkage editor constraints, the text records might also be reblocked.
	Earlier versions of the linkage editor	A load module without count values. Depending on the JCL used and other linkage editor constraints, the text records might also be reblocked.
	The ALTERMOD or COPYMOD functions of the IEBCOPY program in MVS/XA DFP Version 1 Releases 1.1 and 1.2 with the required PTF or MVS/XA DFP Version 2 Release 1.0	A load module with count values inserted. If COPYMOD is used, the module will also be reblocked.
	The ALTERMOD or COPYMOD functions of the IEBCOPY program in MVS/370 DFP Release 1.1	A non-overlay load module has count values inserted; an overlay module does not. If COPYMOD is used, a non-overlay module will also be reblocked.
	Earlier versions of IEBCOPY	A load module without count values. The module is not reblocked.
	MVS/XA program fetch	In some cases, you might observe program fetch performance degradation.
	MVS/370 program fetch	No change.
A load module with count values	MVS/XA DFP linkage editor MVS/370 DFP linkage editor DFDS 1.4 linkage editor with the required PTF installed	A load module with count values inserted. Depending on the JCL used and other linkage editor constraints, the text records might also be reblocked.
	Earlier versions of the linkage editor	A load module without count values. Depending on the JCL used and other linkage editor constraints, the text records might also be reblocked.
	The ALTERMOD or COPYMOD functions of the IEBCOPY program in MVS/XA DFP Version 1 Releases 1.1 and 1.2 with the required PTF or MVS/XA DFP Version 2 Release 1.0	A load module with count values inserted. If COPYMOD is used, a module will also be reblocked.
	The ALTERMOD or COPYMOD functions of the IEBCOPY program in MVS/370 DFP Release 1.1	A non-overlay load module has count values inserted; an overlay module does not. If COPYMOD is used, a non-overlay module will also be reblocked.
	Earlier versions of IEBCOPY	The count values remain. The module is not reblocked.
	MVS/XA program fetch	Depending on the text record lengths, program fetch might perform at its best.
	MVS/370 program fetch	No change.

Figure 8-1. Processing Load Modules

Factors Affecting Text Block Sizes

Several factors affect how the linkage editor determines text block sizes:

- The REGION parameter on the JOB and EXEC JCL statements and on the EXEC statements in SYS1.PROCLIB. The default REGION values defined for the installation can also affect the text block size.

- The values specified for the 'SIZE=(value1,value2)' parameter on the LKED EXEC statement. The SIZE values specify the amount of virtual storage the linkage editor is to use.

- The block size of the previously allocated output library identified on the SYSLMOD DD statement.

- The DCBS option on the PARM parameter of the LKED EXEC statement. Using DCBS allows you to override the block size originally specified for the output data set.

- The block size of the intermediate data set (the data set named on the SYSUT1 DD statement). The linkage editor determines the intermediate data set's block size based on several factors, including the device type.

- The block size of the:

  - Primary input data set (named on the SYSLIN DD statement)
  - Automatic call library (named on the SYSLIB DD statement)
  - The diagnostic output data set (named on the SYSPRINT DD statement)

- The DC option on the LKED EXEC statement. DC causes the linkage editor to construct text blocks of 1K or less.

- The sizes of the control sections (CSECTs) and named common areas being combined into one load module. When building a text record, the linkage editor puts multiple CSECTs and named common areas into the same record, until it runs into a CSECT or named common area that does not completely fit. The linkage editor then truncates that text record and begins a new one. It never splits CSECTs or named common areas across text records that contain other CSECTs or named common areas.

  That restriction also applies if a CSECT or named common area is larger than the maximum text block allowed. The linkage editor does not put any other CSECT or named common area in the last text record occupied by the large CSECT or named common area. Because of this restriction, text records are not always uniform in size or as large as the linkage editor allows for the output device.

The linkage editor uses all of these controls to determine the maximum block size. Poorly chosen values can force the linkage editor to build text blocks smaller than necessary. Therefore, you need to carefully consider any that you specify. The *Linkage Editor and Loader User's Guide* describes how the linkage editor determines block size and how you can control it.

Following are some of the more common reasons less-than-optimal block sizes are produced:

- The REGION value is too small. Unlike the MVS/370 linkage editor, the MVS/XA version does not execute in an overlay environment. Therefore, it requires 32K more virtual storage than the MVS/370 linkage editor. Because of the additional storage requirement, the link edit step might fail or the linkage editor might be forced to build smaller text blocks than would the MVS/370 linkage editor.

- The values specified on the SIZE parameter cause an inadequate output buffer length.

- The intermediate data set (SYSUT1) supports a smaller maximum record length than the output data set (SYSLMOD).

- The data set was copied from a different type of device and not link edited again. For example, a data set was copied from a 3330 device to a 3350 device. (Text records cannot exceed 12 K on 3330 devices; 3350 devices allow 18 K records.)

- The SYSLMOD DD statement specifies a less-than-optimal BLKSIZE value.

Using a New Directory for LNKLST Data Sets

A new LNKLST lookaside (LLA) function in Release 2.1.1 creates and maintains a directory of modules in the LNKLST concatenation. BLDL can use the new directory instead of the PDS directories or the BLDL table to locate modules in the LNKLST concatenation. Because the new directory is hashed and resides in the new LLA address space, using it has several advantages:

- You no longer need to tune the LNKLST concatenation for optimal performance, nor do you need to maintain BLDL lists. The order in which data sets are concatenated does not affect the time required to search hashed directories. Because the new directory is in storage, BLDL lists are unnecessary.

- The new directory eliminates the channel and device contention that occurs when searching PDS directories.

- Data sets in the LNKLST concatenation no longer have to be APF authorized. Consequently, you can include unauthorized data sets formerly included in STEP, JOB, and TASK libraries. The procedure for including unauthorized data sets is described later.

- The LNKLST concatenation can include up to 123 data sets. Earlier MVS releases allow a maximum of 16.

- You can update the LLA directory without performing an IPL. Adding or changing entries in the BLDL table requires an IPL. New commands for updating the directory are described later.

- You can control the amount of paging done for the LLA directory by putting the LLA address space in a separate SRM performance group and adjusting its working set size.

Starting the LLA Function

You might not have to do anything to start the LLA function. The IEACMD00 PARMLIB member shipped with Release 2.1.1 contains a new command, START LLA, which starts a new LLA procedure in SYS1.PROCLIB. The new procedure, in turn, causes the system to build and begin using the LLA directory.

If you use an IEACMDxx member other than IEACMD00, ensure that it includes a START LLA command. Also ensure that the SYS1.PROCLIB data set you use includes the LLA procedure. In addition, if you want to include unauthorized data sets in the LNKLST concatenation, you must specify the new LNKAUTH system parameter, as described in the following topic. The default is to treat all modules fetched via the LNKLST concatenation as APF authorized, which is consistent with earlier releases.

Including Data Sets that Are Not APF Authorized

The new LNKAUTH system parameter has two values:

LNKAUTH=APFTAB - The system treats only those data sets named in the APF table as APF authorized.

LNKAUTH=LNKLST - The system treats all data sets in the LNKLST concatenation as APF authorized, regardless of whether their names are in the APF table. LNKLST is the default.

To include data sets that are not APF authorized in the LNKLST concatenation:

- Either include LNKAUTH=APFTAB in the appropriate IEASYSxx PARMLIB member or have the operator specify it when prompted for system parameters.

- Include in the APF table all LNKLST data sets to be APF authorized.

Note that the APF authorization established at IPL time remains in effect for the duration of the IPL, even if the LLA function is stopped.

Updating the LLA Directory

To add or change an entry in the new directory, either:

- Issue a MODIFY LLA,REFRESH command to refresh the directory.

- Stop the LLA function by issuing the STOP LLA command, then build a new directory by issuing a START LLA command.

Unless your installation shares LNKLST data sets among multiple systems, the first method is preferable. The system can refresh the directory without interrupting its use. Stopping the LLA function causes BLDL to search the PDS directories instead of the LLA directory, which can degrade performance.

If more than one system shares the LNKLST data sets, the second method might be better. It allows you to synchronize directory updates. Operators stop the LLA function on all systems, then restart it via the LLA START command.

SMF Data Set Placement

If the device on which an SMF data set resides requires intervention, SMF can generate a large backlog of records while the device is unavailable. Changes in Release 1.1 can alleviate the backlog, provided you do not put all SMF data sets on the same device. If a system with Release 2.1.1 installed detects that SMF is not writing from buffers, it attempts to use another SMF data set. However, moving to another data set solves the problem only if that data set is on another device.

Using Residency Time to Calculate the Page-in Rate of an Address Space

If your installation is at the Release 2.1.2 level, you can request that SRM use residency time instead of execution time when calculating the page-in rate for address spaces in a specified performance group. However, SRM continues to base the page-in rate for cross memory address spaces on elapsed time. Previously, the system used execution time only, except in the case of cross memory address spaces.

Basing the rate on residency time allows the system to decrease the target working set size of an address space while the address space is inactive. Because most installations prefer to maintain minimum working sets for swappable address spaces, requesting residency time calculations is an option mainly for address spaces that are non-swappable.

Basing the rate on execution time protects the frames in the working set while the address space is inactive. The system adjusts the target working set size only while the address space is active. While the address space is inactive, the target size remains the same as when last calculated.

To request that SRM use residency time, use the new IPS parameter, PPGRTR. PPGRTR specifies the high or low limit the rate must exceed before SRM adjusts the address space's working set size. Previously, PPGRT and CPGRT were the only parameters for specifying page-in thresholds.

Changes to ASM's Paging Algorithms

The Release 2.1.2 level of ASM uses different algorithms for selecting local page data sets and slots on page data sets. The changes are designed to make the paging process more efficient and might result in less tuning effort on your part.

Changes to the Data Set Selection Algorithm

The new data set selection algorithm distributes paging I/O more evenly among local page data sets. ASM continues to maintain the same three circular queues of control blocks representing local page data sets: one for local page data sets on cached auxiliary storage subsystems, one for data sets on fixed-head devices, and one for data sets on movable-head devices. As before, ASM tries to write first to a data set on a cached auxiliary storage subsystem. However, instead of

picking the next available data set that contains free space, ASM now also considers the responsiveness of the device and might avoid unresponsive data sets.

ASM begins searching at the data set following the one last selected from the queue. ASM considers each data set on one queue before continuing to the next queue.

Changes to the Slot Selection Algorithm

The new slot selection algorithm tries to reduce device arm movement and seek time by concentrating paging I/O toward the front of the data set.

Resource Access Control Facility (RACF) Considerations

The Resource Access Control Facility (RACF), beginning with Version 1 Release 6, includes features to improve performance on systems using RACF. Some of the most useful are:

- Global access checking (GAC)
- The resident index and data blocks options
- Generic profiles

Global access checking bypasses normal RACHECK processing and provides a fast path to specific levels of access to data sets by means of generic naming conventions. For example, with global access checking, an entry in the RACF Global Access Checking Table like

```
SYS1.*/read
```

permits reading of all data sets with names that begin with SYS1.

With the resident index and data blocks options a portion of the common service area (CSA) of the virtual address space is reserved for heavily used portions (blocks) of the RACF data set(s). Depending upon RACF activity, the most recently used RACF index and data blocks can reside in storage. Similarly, the generic profile checking facility allows RACF SVCs to build resident profiles within the local system queue area (LSQA) that each protect several resources. Having these security data resident, reduces the number of I/O requests to RACF data set(s) and speeds access to RACF protected resources.

The *RACF Security Administrator's Guide* and the *SPL: RACF* give details on using global access checking, generic profiles, and, resident data blocks.

Managing Contention for Processors with the Vector Facility

A processor complex with the Vector Facility installed on a subset of the processors can function as an asymmetric processing complex. Some jobs run only on the processors with the Vector Facility and other jobs run on any processor. The operating system manages the Vector Facility resource to ensure that jobs assigned to the Vector Facility have access to it. It divides the workload for other jobs among all of the processors in the complex.

In such an asymmetric environment, the queue for the Vector Facility may need to be controlled. Because the system resources manager (SRM) manages the multiprogramming level so that the total processor resource is not overutilized, it may consider a complex to be in balance even though the processors with the Vector Facility is overutilized and the other processors are underutilized. When this imbalance occurs, the installation can manage contention for the Vector Facility by using the normal tuning controls, such as setting initiator limits and limits on the multiprogramming level for the domain in which the Vector Facility jobs execute.

Automatic Priority Group (APG) Specifications

As of Release 2.1.7 it is no longer necessary to use a rotate priority group specification (Rx) to ensure that equal priority address spaces will get their fair share of the processor resource. Instead, the system dispatches address spaces of equal priority in the order in which they become ready. This dispatching technique means that, as of Release 2.1.7, existing rotate priority specifications have no significance. When the system encounters an Rx specification in the IEAIPSxx member of SYS1.PARMLIB, the system accepts it as Fx, the first fixed priority.

In the past, each set of sixteen priorities in the APG was divided into three groups corresponding to each of the three priority control algorithms. The first 10 priorities (0 - 9) were assigned to the mean-time-to-wait algorithm, the eleventh (A) was assigned to the rotate algorithm, and the last 5 (B - F) were assigned to the fixed algorithm.

Release 2.1.7 removes the rotate algorithm and adds the eleventh priority to the fixed priorities. There are now 6 fixed priorities (A - F) specified as Fx, Fx0, Fx1, Fx2, Fx3, and Fx4, respectively.

Chapter 9. Coexistence Considerations

Running both MVS/370 and MVS/XA in the same installation is referred to as coexistence. Installations maintain coexistence because they:

- Have processors that support only MVS/370
- Must use one type of operating system as backup for the other

Most installations will maintain some form of coexistence during the migration period. Many will continue to run both operating systems for some time after MVS/XA is established as a production system.

MVS/370 and MVS/XA can coexist either as independent operating systems running on different processors, as independent operating systems that alternately run on the same processor, or as loosely-coupled operating systems.

In all types of coexistence, the major objectives are to:

- Maintain programs that can run on either system.
- In some cases, ensure that MVS/370 can run the MVS/XA workload or that MVS/XA can run the MVS/370 workload in backup situations.

When MVS/XA and MVS/370 systems are loosely-coupled, installations have some additional considerations, including:

- Ensuring that jobs that must run on a particular system are routed to that system
- Determining which data sets can be shared
- Reviewing dynamic system interchange (DSI) procedures

This chapter includes information related to these topics.

Note: Further coexistence considerations exist for installations running more than one release of JES2 or JES3 in the same processor complex. See the *JES3 Conversion Notebook* and the *MVS/XA JES2 User Modifications and Macros* for an explanation of JES coexistence considerations.

Maintaining Programs that Can Run on Both MVS/370 and MVS/XA Systems

Topics in this section describe:

- Instructions for assembling and link editing programs that must run on both MVS/370 and MVS/XA systems
- Criteria for ensuring that programs can run on both systems
- Ways to avoid unnecessary 24-bit dependencies in new programs
- Instructions for using the SPLEVEL macro to generate compatible expansions of fourteen downward incompatible macros
- Two methods of ensuring that programs using the SYNCH macro can run on both MVS/370 and MVS/XA systems

Assembling and Link Editing Programs

In a mixed installation, use Assembler H Version 2 to assemble all programs that use new 370-XA instructions or that are to be run in 31-bit addressing mode. Use the linkage editor in either DFDS Release 1.4 (with the PTF for APAR OZ57635 installed), MVS/XA DFP, or MVS/370 DFP to link edit programs that will be run on the MVS/XA system in *24-bit* addressing mode. Using one of those linkage editors is important for fetch performance reasons, as described in Chapter 8. Use the MVS/XA DFP or MVS/370 DFP linkage editors to link edit programs that are to be run in *31-bit* addressing mode.

The MVS/XA DFP and MVS/370 DFP linkage editors are the only ones that insert AMODE and RMODE indicators in CESD entries for load module CSECTs and in the partitioned data set (PDS) entries for load modules. The linkage editors in OS/VS2 MVS and DFDS do not support AMODE and RMODE indicators.

- They do not insert or retain the AMODE and RMODE indicators in the PDS directory entry. If a load module is link edited using one of those linkage editors, any AMODE or RMODE indicators already in the PDS directory entry are deleted.
- The same linkage editors ignore any AMODE or RMODE indicators in the ESD or CESD. Indicators already present remain unchanged. New ones are not inserted. (Assembler H Version 2 and selected HLL compilers, not the linkage editor, insert AMODE and RMODE indicators in the ESD entries of object modules.)

"Establishing a Program's Addressing Mode" in Chapter 3 describes how AMODE and RMODE indicators are inserted and used in more detail.

Guidelines for Ensuring Program Compatibility

If a program is to run on both MVS/370 and MVS/XA systems, the program must:

- Perform the desired function on both systems. A program might execute without error on both systems, but not produce the desired results (for example, an SMF post processor that analyzes data that has different formats in MVS/XA and MVS/370).

- Use the MVS/370 expansion of macros whose MVS/XA expansions do not work on MVS/370 systems. There are fourteen such macros. "Handling Downward Incompatible Macros" lists them and describes ways of obtaining the appropriate expansion. In some cases, the SYNCH macro is also downward incompatible. See "Downward Incompatible SYNCH Macros" for details.

- Not use new MVS/XA function. New function includes new instructions, new macros, and new parameters, keywords, or options on existing macros. Exceptions are the new LOC, VRC, and VRU parameters on the GETMAIN macro and the AMODE=24 parameter on SYNCH. Those parameters generate object code that works on MVS/370 systems. See "Parameters on the GETMAIN Macro Instruction" in Chapter 3.

- Use only system services that are supported in both MVS/370 and MVS/XA. Chapter 3 identifies functions not supported in MVS/XA.

- Provide dual paths for functions that are not compatible between MVS/370 and MVS/XA and dynamically select the proper path at execution time. Bit 0 in the CVTDCB field of the CVT indicates whether MVS/XA is executing. (If it is, bit 0 equals 1.) The MVS/XA CVT map defines the bit as CVTMVSE. Method 3 in "Handling Downward Incompatible Macros" shows the dual path section of a sample program.

If the program uses non-standard interfaces to system modules or uses system control blocks, you must also ensure that:

- Methods of invoking system services work in both MVS/370 and MVS/XA.
- Control block references can be used in both MVS/370 and MVS/XA.

Some programs require different versions to run in MVS/370 and MVS/XA (for example, RMF analysis routines). Installations can either:

- Keep each version of the program in a separate library.

- Keep both versions of the program in the same library, but give each a different name.

- Rewrite the program so that it has dual paths and dynamically selects the proper path at execution time, as mentioned earlier.

When designing new programs that must run on both MVS/370 and MVS/XA systems, avoid unnecessary 24-bit dependencies in programs that might be executed in 31-bit addressing mode:

- Use fullword address fields, even if the value in the field is below 16 megabytes.
- Avoid using the load address (LA) instruction to clear the high-order byte. In 31-bit addressing mode, the LA instruction clears only the high-order bit, not the entire byte, as it does in 24-bit addressing mode.
- When coding BAL or BALR, avoid using the information saved in the high-order byte of the first operand (the instruction length code, program mask, and condition code). When executed in 31-bit addressing mode, BAL and BALR do not save that information. 370-XA processors provide a new instruction, IPM (Insert Program Mask), which saves the program mask and condition code when executing in 370-XA mode.
- Use ESTAE instead of STAE. STAE is not changed to support 31-bit addressing.
- When obtaining large amounts of storage, use the VRU, VRC, RU, and RC forms of GETMAIN and FREEMAIN. These forms support a new LOC parameter, which allows users to specify from where virtual storage is to be obtained and how it is to be backed when fixed. See "Parameters on the GETMAIN Macro Instruction" in Chapter 3.

  In contrast, VSM satisfies the LC, LU, VC, VU, EC, EU, and R forms of GETMAIN requests with virtual storage below 16 megabytes. Also, when fixing storage obtained via those forms of GETMAIN, RSM always uses real storage below 16 megabytes.

The following macros are not changed to provide full 31-bit support. MVS/XA provides new services instead. You might want to use dual paths when using any of these services:

- PGFIX, PGFREE, PGRLSE, PGLOAD, and PGOUT. A new PGSER macro provides the equivalent services and supports 31-bit addresses.
- SPIE. ESPIE is the MVS/XA counterpart.

Programs that Run in System 370, 370-XA 31-Bit, or 370-XA 24-Bit Addressing Modes

There are two recommended methods for writing programs that will run in System 370 mode, 370-XA 31-bit addressing mode, and 370-XA 24-bit addressing mode. They each allow you to save and restore the program mask and condition code. Note that these methods assume the programs will run under Assembler H which is required for executing the IPM instruction.

Method 1 - For systems running MVS/SP Version 1 Release 3.0 with the fix for APAR OZ62229 installed or for a later release:

At initialization time your application can interrogate the CVTDCB field of the CVT control block to determine what level system it is running under. If it is

370-XA (24 or 31-bit addressing mode), you can move a predefined IPM instruction to an area that can be "globally" accessed by tasks in the application's address space and execute the IPM instruction by means of the EX instruction. If the system is running in 370 mode, you can move a BALR instruction to this area. Both IPM and BALR return the condition code in bit positions 2 and 3 and the program mask in bit positions 4 through 7 of the user-specified register. See the example program segment that follows.

Using Predefined IPM and BALR Instructions

```
1 INIT      CSECT
2 *          .
3 *          .
4           BALR  3,0
5           USING *,3
6 *
7           USING CVT,2
8           LA    2,16
9           TM    CVTDCB,CVTMVSE
                                *Running under MVS/XA
10          BO    XA
                                *Perform MVS/XA processing
11 *
12 *      NOT RUNNING UNDER MVS/XA
13 *
14          MVC   EXAREA(BALRLN),BALRINST
                                *Move BALR instruction
15 *                            *to commonly addressable
16 *                            *execute area
17          B     BYPASSXA      *Bypass MVS/XA processing
18 *
19 *     MVS/XA PROCESSING
20 *
21 XA       EQU   *
22          MVC    EXAREA(IPMLN),IPMINST
                                *Move IPM instruction
23 *                            *to commonly addressable
24 *                            *execute area
25 BYPASSXA EQU *
26 *          .
27 *          .                 *Execution continues
28 *          .
29 BALRINST EQU *
30 BALR     5,0
                                *BALR using R5
31 BALRLN   EQU *-BALRINST
                                *Length of data to move
32 *
33 IPMINST  EQU *
34          IPM 5               *IPM using Reg 5
35 IPMLN    EQU *-IPMINST
                                *Length of data to move
36 *
37 EXAREA   DS  XL4
38 *
39          CVT DSECT=YES
```

Code executed after initialization

```
            LA  6,0             *Clear work register
            EX  6,EXAREA        *Execute correct instruction
                                *Program mask/cc returned
                                *in register 5
```

Method 2 -For any MVS System:

You can use the facts that the BALR instruction is a halfword in length (ILC is b'01'), that in 370 and 370-XA 24-bit addressing modes, BALR sets the high order bit of the return register to 0, and that in 370-XA 31-bit addressing mode, BALR sets this bit to 1. First, you can retrieve and save the condition code. Then you can issue a BALR instruction and check the high order bit of the return register. If it is 1, the system is running in 31-bit addressing mode and you must send control along a programming path that uses the IPM instruction. If the bit is 0, the system is not in 31-bit addressing mode and you can isolate and retrieve the program mask made available by the BALR. See the example program segment that follows:

Using BALR/IPM to Save the Condition Code and Program Mask

```
 1 INIT     CSECT
 2 *          .
 3          USING *,15
 4 *
 5          LA     6,0                 *Clear a work reg to zero
 6 *                                   *Note: CC not changed
 7          BC     0,CONTINUE          *Is condition code = 0?
 8          BC     1,CC1               *Is condition code = 1?
 9          BC     2,CC2               *Is condition code = 2?
10 *                                   *Default is CC = 3
11 *
12          ICM    6,B'1000',MASKCC3 *Mask for CC=3 in
13 *                                  *bit positions 2 through 3
14          B      CONTINUE
15 *
16 CC1      EQU    *
17          ICM    6,B'1000',MASKCC1 *Insert mask for CC=1 in
18                                    *bit positions 2 through 3
19 *
20          B      CONTINUE
21 *
22 CC2      EQU    *
23          ICM    6,B'1000',MASKCC2 *Insert mask for CC=2 in
24                                    *bit positions 2 through 3
25 *
26CONTINUE EQU    *
27          BALR   5,0
28          LTR    5,5                 *Determine addressing mode
29          BP     T24BIT              *If positive, 24-bit
30                                     *addressing mode
31          IPM    5                   *Get program mask, CC
32 *
33T24BIT    EQU    *
34          N      5,=X'0F000000'      *Isolate program mask
35          OR     6,5                 *Reg 6 now has condition code
36            .                        *and program mask
              .                        *Execution continues
MASKCC1     DC     B'00010000'
MASKCC2     DC     B'00100000'
MASKCC3     DC     B'00110000'
            LTORG
                 =X'0F000000'
            END
```

Handling Downward Incompatible Macros

Most of the MVS/XA expansions of previously existing macro instructions run on both MVS/370 and MVS/XA systems (that is, the macro instructions are downward compatible). The following macro instructions are exceptions. The MVS/XA expansions of these macros will not run on an MVS/370 system.

ATTACH
CHKPT
CLOSE*
ESTAE
EVENTS
FESTAE
INTSECT
OPEN*
SCHEDULE SCOPE=GLOBAL
SDUMP if it specifies new parameters
SETFRR INLINE=YES
SETLOCK RELEASE,TYPE=(reg)/ALL
SMFEXIT
STAX
STIMER
SYNCH, unless it specifies the parameter AMODE=24
WTOR

*** Refer also to "OPEN and CLOSE Requirement for Assembler H Version 2" on page 9-10 and "OPEN and CLOSE and MODE=31" on page 9-10 for further information on the use of these macros.**

To share user-written programs among MVS/370 and MVS/XA systems and to have backup capability while migrating to MVS/XA, users must be able to override the downward incompatible macro expansions with macro expansions that will run on both MVS/370 and MVS/XA systems. MVS/XA provides that capability for all of the macros listed, except SYNCH. (See "Downward Incompatible SYNCH Macros" for instructions on maintaining SYNCH compatibility.)

The MVS/XA MACLIB contains two different expansions for all of the above macros, except SYNCH: an MVS/SP Version 1 Release 3 expansion and an MVS/XA expansion. Note that the source statements that invoke the macro instructions remain the same, only the expansions are different for the two environments. The Version 1 expansions run on both MVS/370 systems and MVS/XA systems executing programs in 24-bit addressing mode. The MVS/XA expansion is required when using any new parameters or options on the above macros. In most cases, the MVS/XA expansion is also required when executing in 31-bit addressing mode. (SCHEDULE, SDUMP, and SETLOCK are exceptions.)

The level of the macro expansion (MVS/370 or MVS/XA) that is generated during assembly depends on the value of an assembler language global SET symbol. When the SET symbol value is 1, the system generates MVS/370 expansions. When the SET symbol value is 2, the system generates MVS/XA expansions.

MVS/SP Version 2 includes a new macro, SPLEVEL, which allows programmers to change the value of the SET symbol. When SPLEVEL itself is assembled, it assigns a value to the SET symbol. That value becomes the default value for the entire installation.

The SPLEVEL macro shipped with MVS/SP Version 2 assigns a SET value of 2. **Therefore, unless a program specifically changes the SET value, the assembler generates MVS/XA macro expansions.**

Your installation can change the SET value shipped with MVS/SP Version 2, or individual programmers can override the SET value in particular programs:

- To change the SET value for the entire installation, after system generation, modify the SPLEVEL source code in SYS1.MACLIB. Change the statement that assigns the SET value: '&DEFAULT SETC n', where 'n' is 1 or 2. Note that when assembling MVS/XA system programs, either at system generation or when applying service, the SET value must be 2. (MVS/XA expansions are required.)

- Programmers can issue within a program the SPLEVEL SET=n macro, where n equals 1 to obtain MVS/370 expansions, or 2 to obtain MVS/XA expansions. The SPLEVEL macro sets the symbol to the specified value for that program's assembly only. Thus, issuing the SPLEVEL macro only affects expansions within the program being assembled. A single program can include multiple SPLEVEL macros to generate different macro expansions.

Obtaining the Appropriate Macro Expansions

Following are three ways programmers can use SPLEVEL to obtain the appropriate macro expansion within their programs. Methods 1 and 2 generate different expansions in different programs (for example, MVS/370 expansions in Program A and MVS/XA expansions in Program B). Method 3 generates different expansions within the same program:

Method 1 - Obtaining different expansions in different programs

Keep the SPLEVEL macro shipped with MVS/SP Version 2 in the SYS1.MACLIB macro library. Put a copy of SPLEVEL into another macro library by itself, and change the source code to establish SET = 1 as the installation default. When assembling programs, use JCL to access the appropriate macro library.

In the following example, the SPLEVEL macro that establishes SET = 1 as the installation default is by itself in the SET1MACS macro library.

To assemble the MVS/XA expansions in programs, use:

```
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
```

To assemble MVS/370 expansions, use:

```
//SYSLIB   DD   DSN=SET1MACS,DISP=SHR
//         DD   DSN=SYS1.MACLIB,DISP=SHR
```

You can, of course, switch the SPLEVEL macros and put the one that establishes SET =1 as the installation default in SYS1.MACLIB.

Method 2 - Obtaining different expansions in different programs

Issue the SPLEVEL SET=n macro once at the beginning of the module to obtain the appropriate expansions:

```
MODULE    CSECT
          SPLEVEL  SET=1
             .
             .
             .
```

Method 3 - Obtaining different expansions within the same program

Assemble both levels of the macro and make an execution-time test to determine which level to execute. The following example invokes the correct level of the WTOR macro:

```
*    DETERMINE WHICH SYSTEM IS EXECUTING
          TM       CVTDCB,CVTMVSE    (CVTMVSE is bit 0 in
          BO       SP2                the CVTDCB field. It
                                      indicates whether
                                      MVS/XA is executing.)
*    INVOKE THE MVS/370 VERSION
*    OF THE WTOR MACRO INSTRUCTION
          SPLEVEL SET=1
          WTOR     .....
          B        CONTINUE
SP2       DS       0H
*    INVOKE THE MVS/XA VERSION
*    OF THE WTOR MACRO INSTRUCTION
          SPLEVEL SET=2
*
          WTOR     .....
CONTINUE  DS       0H
```

Determining Which Level of the Macro Instruction to Use

- You can use a SET value of 2 when you assemble programs that will be run only on MVS/XA systems, regardless of whether they will run in 24- or 31-bit addressing mode.

- Programs that must run on both MVS/370 and MVS/XA systems must assemble at least MVS/370 macro expansions. (Programs with dual paths assemble the MVS/XA expansions as well.)

- Programs that are designed to run on both MVS/370 and MVS/XA systems and that require MVS/XA expansions on the MVS/XA system must obtain both expansions and determine at execution time which level to execute. (See Method 3 in the previous examples.)

- When assembling MVS/XA system programs, either at system generation or when applying maintenance, the SET value must be 2 (that is, MVS/XA expansions are required).

- Programs that run on MVS/XA systems in 31-bit addressing mode must use the MVS/XA expansions of the following downward incompatible macros:

ATTACH	FESTAE	STAX
ESTAE	INTSECT	STIMER
EVENTS	SMFEXIT	WTOR

- Programs that specify any new parameters, keywords, or options on the macros must use the MVS/XA expansions.

OPEN and CLOSE Requirement for Assembler H Version 2

MVS/XA DFP Version 2 Release 3.0, introduces an additional situation requiring the use of Assembler H Version 2. As of this release, the use of either the OPEN or CLOSE macro requires, by default, the use of Assembler H Version 2.

Beginning with MVS/XA DFP Version 2 Release 3.0, both OPEN and CLOSE test the SPLEVEL SET value when they are expanded:

- If SPLEVEL SET=2 is coded (which is the default for SPLEVEL), when both OPEN and CLOSE are expanded they generate a BAS instruction. This requires assembly with Assembler H Version 2.

- IF SPLEVEL SET=1 is coded, when both OPEN and CLOSE are expanded they generate a BAL instruction. This does not require assembly with Assembler H Version 2.

Note: Some, but not all, of the processors that support MVS/370 also support the BAS instruction.

OPEN and CLOSE and MODE=31

If MODE=31 is coded on either the OPEN or CLOSE macros, the expansions of the macros will contain a new, long form of the parameter list passed to OPEN or CLOSE, and the programs will execute only on MVS/XA DFP Version 2 Release 3.0 or a later release. In addition, MODE=31 is appropriate only when the SET value is 2. The system treats the combination of a SET value of 1 and either OPEN MODE=31 or CLOSE MODE=31 as an error.

EOV and MODE=31

If MODE=31 is coded on the EOV macro, the program will execute only on MVS/XA DFP Version 2 Release 3.0 or later. Do not code MODE=31 on the EOV macro if the program must execute on earlier levels of MVS/XA DFP or in any other environment which has no support for EOV MODE=31.

Downward Incompatible SYNCH Macros

In some cases, programs that use the SYNCH macro and are assembled using the MVS/XA MACLIB will not run on an MVS/370 system. The downward incompatible programs are those that do NOT specify the parameter AMODE=24 on SYNCH (that is, programs that either omit the AMODE parameter or specify AMODE=31, AMODE=DEFINED, or AMODE=CALLER).

Unless SYNCH specifies AMODE=24, after assembly, the SYNCH macro expansion contains a nonzero value in a previously reserved field in the SYNCH parameter list. The MVS/XA expansion uses that field as an AMODE indicator. The MVS/370 SYNCH processor treats the nonzero field as an error and issues ABEND x'206'.

If a program that issues SYNCH must run on both MVS/XA and MVS/370 systems, you need to ensure that the AMODE indicator field in the SYNCH parameter list is zero. You can either:

- Use the MVS/370 MACLIB when assembling the program.
- Specify the parameter AMODE=24 on the SYNCH macro and use the MVS/XA MACLIB for assembly.

Backup Considerations

Following are backup considerations for installations that must use an MVS/370 system as backup for an MVS/XA system, or use an MVS/XA system as backup for an MVS/370 system:

- For program products that have separate licenses for MVS/370 and MVS/XA, installations need both licenses for backup capability. Such programs include:
  - RMF Version 2 and RMF Version 3
  - MVS/SP Version 1 Release 3 or later releases and MVS/SP Version 2
- User-written programs that access system control blocks or that use authorized services and interfaces might not run on both MVS/370 and MVS/XA systems. If such programs are not compatible between systems, the installation is without backup capability.
- The size of the available private area increases in MVS/XA. Installations that use the additional private area will have to reevaluate the capability of using an MVS/370 system for backup.
- When using a backup processor, installations need to ensure that the backup system can use the current system's IOCDS. The 370/370-XA IOCP can create an IOCDS that can be used in either System/370 or 370-XA mode. See "Creating or Modifying an IOCDS Using the MVS Version of IOCP" in Chapter 2 for more information.
- Installations need to have a procedure defined for changing processor modes and for changing the IPL volume.
- On the MVS/370 system, use DFDSS or an equivalent product instead of IEHDASDR or the DASDR licensed product to produce backup tapes that might need to be restored on an MVS/XA system. Neither IEHDASDR nor the DASDR licensed product are available in MVS/XA. You must use DFDSS or an equivalent function to perform dump/restore operations in MVS/XA. DFDSS does not support the dump format that IEHDASDR or the DASDR licensed product produces.

DFDSS Version 2 (available for Release 2.1.2 and later systems) is upwardly compatible with DFDSS Version 1 functions, and also includes functions not available in DFDSS Version 1. The added functions create differences you may need to be aware of.

DFDSS Version 2, for example, processes data sets as logical entities which allows them to be transferred from one type of device to a different device type. The system version of DFDSS Version 2 restores from logical data-set dumps, but no release of DFDSS Version 1 can restore from this dump format. Similarly, when a logical data-set dump tape has been created by the system version of DFDSS Version 2, the stand-alone version will not restore from that tape. *Device Facility Data Set Services: General Information* GC26-4123 describes the differences between the DFDSS versions.

- MVS/XA can support a larger workload than MVS/370. An MVS/370 system might not be able to support the MVS/XA workload.

Routing Jobs in a Mixed JES2 or JES3 Complex

When MVS/370 and MVS/XA systems are loosely-coupled, installations must ensure that JES routes jobs that must run on a particular system to that system (for example, jobs that use new MVS/XA function).

If a job needs a device that is attached to only one processor, JES3 automatically routes the job to that processor. To ensure the proper job-system match in other situations, installations and programmers can use existing job routing procedures:

- An installation can define specific execution job classes for jobs that must run on MVS/XA, for jobs that must run on MVS/370, and for jobs that can run on either system. The installation then associates each job class with the appropriate processor. Users ensure that their jobs run on the appropriate system by specifying the CLASS= parameter on the job's JCL. JES2 users specify the CLASS= parameter on the JOB statement. JES3 users specify it on the //*MAIN or //JOB statement.

  Note: Routing by job class works only in situations where processors are always running the same operating systems when the job routing takes place. For example, if Job A specifies CLASS=XA, the processor associated with class XA must always be running MVS/XA when Job A executes.

- If a user knows which operating system is running on a processor, the user can specify on the job's JCL the processor on which the job is to run. JES2 programmers use the SYSAFF parameter on the /*JOBPARM statement. JES3 programmers use the SYSTEM parameter on the //*MAIN statement.

TSO users that require a particular system must log on and submit started tasks to that system.

See *SPL: JES2 Initialization and Tuning* or *SPL: JES3 Initialization and Tuning* for more information.

Using Global Resource Serialization

The global resource serialization components of MVS/SP Version 2 and MVS/SP Version 1 Release 3 and later releases are compatible. Therefore, loosely-coupled MVS/XA and MVS/370 systems can use global resource serialization to control data sharing. The RESERVE/DEQ functions are also compatible.

In general, the fact that a global resource serialization complex includes mixed systems does not impose additional restrictions on the types of data sets that can be shared. Depending on the level of the systems in the complex, you might need to modify the RNLs to ensure that VSAM data sets are shared properly. See "Serializing VSAM Data Sets" in Chapter 5 for more information. Also read "Updating SYSTEMS Exclusion RNLs" in the same chapter. *Global Resource Serialization* describes data set sharing in general.

MVS/XA systems and MVS/370 systems that have MVS/370 DFP installed can share VSAM and CVOL catalogs. If Data Facility Extended Function (5740-XYQ) is also installed on the MVS/370 system, the systems can share ICF catalogs as well.

Global Resource Serialization implemented the enhancements described in "Keeping RNLs in GRSRNLxx PARMLIB Members" on page 2-18 and "Using Default RNLs" on page 2-28 beginning with Release 2.1.2 and MVS/SP Version 1 Release 3.5. Loosely-coupled processor complexes will have different start-up operations for Global Resource Serialization if they combine earlier versions with versions of Global Resource Serialization available with these or subsequent releases.

System Data Sets that Cannot be Shared

MVS/XA and MVS/370 systems cannot share the following system data sets: SYS1.LINKLIB, SYS1.LPALIB, SYS1.NUCLEUS, and SYS1.SVCLIB. MVS/370 systems cannot use the MVS/XA SYS1.PARMLIB data set as shipped. Installations also need two versions of SYS1.MACLIB. The MVS/XA system requires the MVS/XA expansions of downward incompatible macros. Also, some mapping macros are unique to MVS/370 or MVS/XA.

Using SYS1.PROCLIB in a Loosely-Coupled JES3 Configuration

Beginning with MVS/SP Version 2 Release 1.2, converter-interpreter (C/I) processing in a loosely-coupled JES3 configuration can take place on more than one processor. When converting jobs that execute procedures, the system performing C/I processing uses the procedures in its own PROCLIB, not necessarily those in the PROCLIB of the system that will run the job.

In a loosely-coupled configuration that includes both MVS/370 and MVS/XA systems, you need to ensure that the processor performing the C/I service uses the procedure appropriate for the system that will run the job. If the procedure for starting a task in MVS/370 is different from the equivalent MVS/XA procedure (as is the RMF procedure), you need to either:

- Modify the procedure to work on both MVS/370 and MVS/XA systems.
- Maintain two procedures and change the name of at least one of them.

"RMF Procedure" in Chapter 2 describes how to create an RMF procedure that starts either RMF Version 2 or Version 3.

DSI Procedures in a Loosely Coupled JES3 Configuration

If one of the processors involved in dynamic system interchange (DSI) is running MVS/XA and the other processor is running MVS/370, you must ensure that:

- The JES3 global function, including a JES3 console, has sufficient devices that are supported by both the MVS/370 and MVS/XA systems. *MVS/XA Conversion Notebook, Volume 2* lists devices supported in MVS/370 but not in MVS/XA.

- Any user-written JES3 routine that must run on a particular system (either MVS/370 or MVS/XA) is disabled before using DSI.

The *JES3 Conversion Notebook* which is available as of MVS/SP Version 2 Release 1. 5 has further DSI information.

Resource Access Control Facility (RACF) Always-Call

The always-call feature that is available beginning with MVS/XA DFP Version 1 Release 2.0 and MVS/370 DFP Release 1.1, ensures that RACF is invoked each time a request to access a data set occurs. In order to provide complete protection, always-call must exist on all MVS systems in the complex that share the RACF protected data. If one system includes always-call and another does not, then RACF security may be bypassed by the users of the system that does not have always-call. The *RACF Security Administrator's Guide* describes always-call.

Resource Access Control Facility (RACF) and VSAM Clusters

Along with the introduction of always-call in MVS/370 DFP Release 1.1 and MVS/XA DFP Version 1 Release 2, RACF simplifies protection of VSAM clusters. There is no longer a need for three RACF profiles (for the cluster name itself, the index component name, and the data component name) since RACF considers that the authority to access the cluster name itself is the same as the authority to access any of the VSAM cluster components.

When sharing VSAM data between an earlier DFP system and a simplified, MVS/370 DFP Release 1.1 or Release 1.2 system, you must protect the data according to the procedure of the earlier system. VSAM data sets protected properly according to the earlier system will be protected on all systems sharing the data. (The additional RACF profiles are ignored and create no interference.) If you do not follow the three-profile procedure the VSAM data sets on the pre-MVS/370 Release 1.1 or Release 1.2 system will not have RACF protection.

Appendix A. Parameter Changes in Incompatible Macros

This appendix describes changes to the parameters that the following macros pass to their service routines:

- ATTACH
- ESTAE
- EVENTS
- SMFEXIT
- STAX
- STIMER
- SYNCH
- WTOR

You need to be concerned about the changes only if you have programs that invoke the associated service routines directly (for example, by branch entry) instead of using the macros. You need to modify the parameter lists that those programs build.

To maintain compatibility, the MVS/XA service routines for all of macros listed except SYNCH accept MVS/370 or MVS/XA parameters. In most cases, the service routines check a flag bit (identified as FLAG BIT in the following figures) to determine which format (MVS/370 or MVS/XA) the input parameters are in. If the bit is 0, the parameters are in MVS/370 format. If the bit is 1, the parameters are either in MVS/XA format or in the format indicated by a format number somewhere in the parameter list. (The only defined format number is 1, which indicates MVS/XA format.)

If you build your own parameters, ensure that the flag bit and, in some cases, the format number correctly specify which version of the parameters you are passing to the service routine.

Note: In the following figures, blank fields represent fields that are not changed.

The FLAG BIT is the high-order bit of byte 8. The FORMAT NUMBER is byte 61.

MVS/370

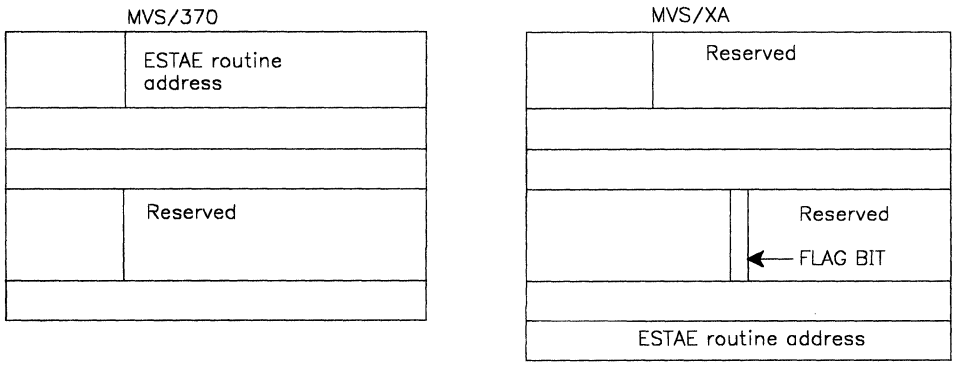
Flags	Entry address	
	DCB address or zeros	
Flags	ECB address	
Flags	Give subpool number or list address	
Flags	Share subpool number or list address	
Flags	End-of-task exit routine address	
Resv.	JSCB address	
Task ID	STAI or ESTAI parameter list address	
Flags	STAI or ESTAI routine address	
Resv.	TASKLIB DCB address	
Flags	Resv.	Length of parameter list
Subpool number(s)		

MVS/XA

Entry address		
DCB address or zeros		
ECB address ← FLAG BIT		
Give subpool number or list address		
Share subpool number or list address		
End-of-task exit routine address		
JSCB address		
STAI or ESTAI parameter list address		
STAI or ESTAI routine address		
TASKLIB DCB address		
Flags	Task ID	Length of parameter list
Subpool number(s)		
Flags	**FORMAT NUMBER**	Reserved

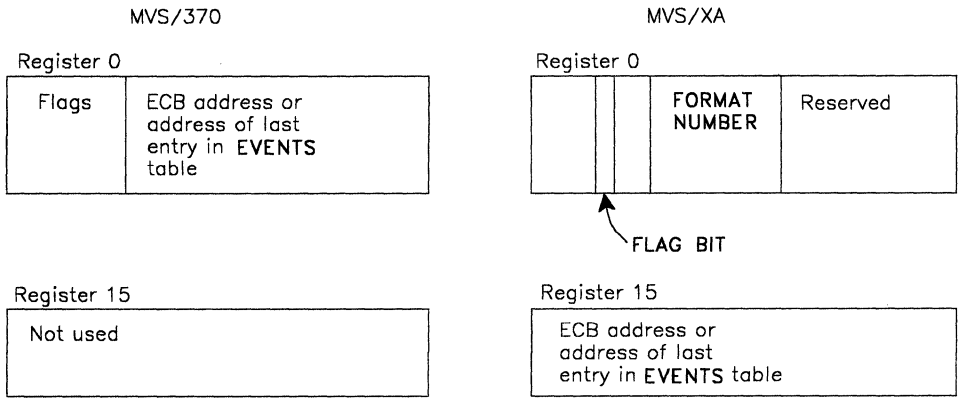
ESTAE Parameter List Changes

The FLAG BIT is the low-order bit of byte 13.



EVENTS Parameter Changes

The FLAG BIT is the fourth bit of byte 0 in Register 0. The FORMAT NUMBER is the second byte of Register 0.

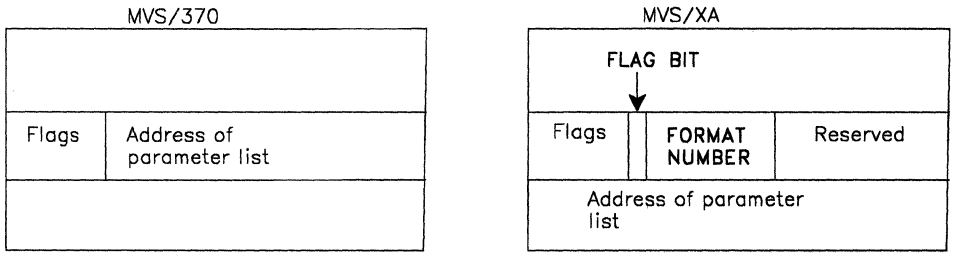


SMFEXIT Parameter List Changes

If the user specifies a work register on the SMFEXIT macro, the macro sets bit six of the parameter list to 1. In MVS/370, the bit is reserved. SMF uses the work register to save and restore the caller's addressing mode.

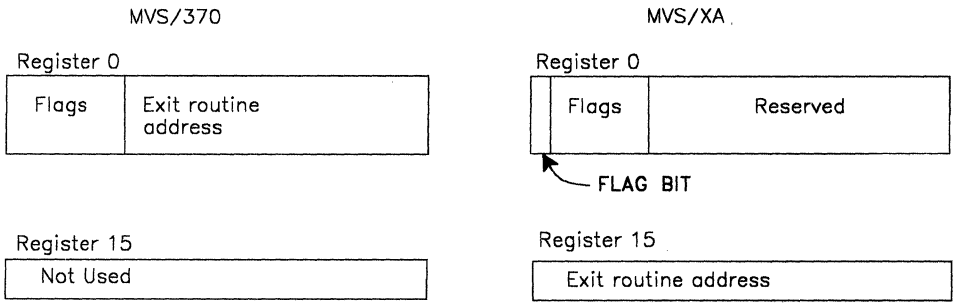
STAX Parameter List Changes

The FLAG BIT is the bit 6 of byte 16. The FORMAT NUMBER is byte 17.

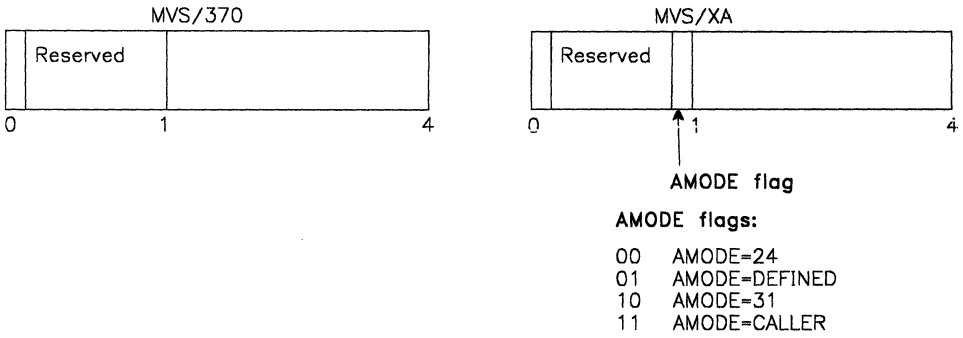


STIMER Parameter Changes

The FLAG BIT is the high-order bit of byte 0.



SYNCH Parameter List Changes



For more information, see "Downward Incompatible SYNCH Macros" in Chapter 9.

WTOR Parameter List Changes

The FLAG BIT is the high-order bit of byte 0.

MVS/370

Reply length	Address of reply buffer
Zero	

MVS/XA

	Address of reply buffer
FLAG BIT	
Reply length	

Appendix B. Control Block Updates

Figure B-1 lists the Release 2.1.x control blocks that are new, updated, or deleted or that can reside anywhere in virtual storage (above or below 16 Mb). Because updates may have occurred to some of the control blocks listed, be sure to consult the corresponding list in the *MVS/XA Conversion Notebook, Volume 2*. It contains the control block updates that have occurred for Release 2.2.x.

The figure that follows does not include control blocks that are not updated and that must reside below 16 Mb. The parentheses contain the name of the mapping macro that defines the structure of the associated control block.

If a control block can reside anywhere, the figure indicates it might be above 16 Mb. If a control block's virtual storage location depends on the caller, the notes column indicates that the location is specified by the user. Note that Release 2.1.7 combines releases 2.1.3VFE and 2.1.3AE.

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
ABDPL (IHAABDPL)	x								x			x	
			x						x			x	
ABEPL (IHAABEPL)	x							x			x		
AE (IHAAE)	x							x			x		
AMDDATA	x								x		x		
		x							x		x		
				x					x		x		
					x				x		x		
AMRQ (IHACTM)		x							x			x	
AQAT (IHAAQAT)	x							x			x		
AQE (IHAAQE)	x									x			Replaced by IHAAE
ASCB (IHAASCB)	x								x			x	
		x							x			x	
				x					x			x	
					x				x			x	
							x		x			x	
ASMHD (ILRASMHD)	x								x		x		
ASMVT (ILRASMVT)	x								x			x	
		x							x			x	
			x						x			x	
ASSB (IHAASSB)					x			x			x		
ASTE (IHAASTE)	x								x			x	
ASVT (IHAASVT)	x								x			x	
			x						x			x	
ASXB (IHAASXB)	x									x			x
ATECB (IEFZB432)		x							x			x	
ATTCH (IEZATTCH)	x								x		x		User-specified
		x							x		x		
AWA (IEFVMAWA)		x							x			x	
BASEA (IEEBASEA)	x								x			x	Resides in the nucleus
		x							x			x	
BEB (IECDBEB)		x							x			x	
BLSRDTDT			x					x				x	
BLSRDUT			x					x				x	
BLSRESSY			x					x			x		
BLSRRDSY			x					x				x	
BLSRSST			x					x				x	
CAT (IECDCAT)	x									x			
CAW	x									x			
CBLS (IHACBLS)					x			x				x	
CCT (IRACCT)	x								x		x		
				x					x		x		
							x		x		x		

Figure B-1 (Part 1 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
CCW (IOSDCCW)	x							x				x	New Format 1 CCW. Format 0 (the MVS/370 format) is unchanged.
CDA (IGFCDA)	x									x			
CDE (IHACDE)	x								x			x	
			x						x			x	
CHRB (IOSDCHRB)			x						x		x		
CHT (IECDCHT)	x									x			
CIB (IEZCIB)	x								x			x	
		x						x			x		
CIFP (IEFCIFP)		x						x			x		
CIMP (IEFVMMWA)		x					x				x		
CLRATT (IEEVC102)	x						x				x		
CMCT (IRACMCT)	x							x			x		In the extended nucleus
				x				x		x			
COMMON	x							x			x		
		x						x			x		
			x					x			x		
COMWA (IEFCOMWA)		x						x			x		
CPT (IECDCPT)	x									x			
CQB (IHACTM)		x							x			x	
CQE (IHACTM)		x							x			x	
CRCA (IECDCRCA)	x									x			
CSCB (IEECHAIN)		x							x			x	
			x						x			x	
CSD (IHACSD)	x								x			x	
				x					x			x	
					x				x			x	
							x		x			x	
CST (IECDCST)	x									x			
CSWK (IECDCSWK)	x									x			
CTM (IHACTM)	x								x				Only the XVSAV and CXSA maps have changed. The COMPL can be above 16 Mb. The others must be below 16 Mb.
CTXT (IEZVX100)			x					x					Mapping macro
CVRWA (IEFCVRWA)		x							x			x	

Figure B-1 (Part 2 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
CVT	x								x			x	
		x							x			x	
				x					x			x	
					x				x			x	The previously reserved field, CVT0FN00, is named CVTVPRM and contains two halfwords with Vector Facility information.
						x			x			x	The previously reserved field, CVTRSV98, is named CVTAVVT and contains availability manager information.
							x		x			x	The Release 2.1.7 CVT combines the 2.1.3VFE and 2.1.3AE changes.
CXSA (IHACTM)	x								x			x	
		x							x			x	
DACB (IKJDACB)		x							x			x	
DALDDNAM (IEFZB4D2)		x							x		x		
DBOX (IOSDBOX)		x							x		x		
			x						x		x		
DCQ (IHADCQ)	x							x				x	
DDP (IGFDDPRM)			x						x			x	
DDR (IHADDR)	x								x			x	
DDT (IECDDT)	x								x			x	
			x						x			x	
DEB (IEZDEB)	x								x			x	
DEVTAB (IFDEVTAB)			x						x			x	
DFE (IHADFE)	x							x			x		
DFLM (ADYDFLM)		x						x				x	
DMDT (IRADMDT)	x										x		
				x					x		x		
DOMC (IHADOMC)	x								x		x		
DOMPL (IHACTM)	x										x		
		x							x		x		
DQE (IHADQE)	x								x		x		
DSAB (IHADSAB)	x								x			x	
			x						x			x	
DSPD (ADYDSPD)		x						x			x		
DSTAT (ADYDSTAT)		x						x				x	
DSVCB (IHADSVCB)	x							x			x		
		x							x			x	

Figure B-1 (Part 3 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
DSX (ADYDSX)		x						x			x		
ECB (IHAECB)	x								x		x		User-specified
ECBE	x								x		x		User-specified
ENFCT (IEFENFCT)	x								x			x	
ENV (IGFENV)	x								x		x		
EPAT (IRAEPAT)				x					x		x		
EPCB (IECDEPCB)		x							x			x	
EPDT (IRAEPDT)				x					x		x		
EPIE (IHAEPIE)	x							x				x	
ERPIB (IGFERPIB)	x									x			
ESPI (IHAESPI)	x							x				x	
ESTA (IHAESTA)	x								x		x		User-specified
ESW (IHAESW)	x								x		x		
			x						x		x		
ETD (IHAETD)	x								x		x		User-specified
ETE (IHAETE)	x								x			x	
EVNT (IHAEVNT)	x								x			x	
EWA (EWAMAP)	x								x			x	
			x						x			x	
FBQE (IHAFBQE)	x								x		x		
FIX (IECDFIX)		x							x			x	
FQE (IHAFQE)	x								x		x		
FRRS (IHAFRRS)	x								x			x	
FTPT (IHACTM)		x							x			x	
GDA (IHAGDA)	x								x		x		
GENX (IEEZB816)			x					x			x		
GETPTWT (IEFZB600)	x									x			Replaced by the VSM parameter list
GTF records:													
CCW	x								x				
EOS	x								x				
IO	x								x				
I/O instructions	x							x					
PCI	x								x				
SIO	x									x			Replaced by new I/O instructions
UIO	x									x			
GSDA (IHAGSDA)			x						x			x	
GVT (ISGGVT)	x								x			x	
		x							x			x	
			x						x			x	
GWT (IHAGWT)	x								x		x		
HCL (IHAHCLOG)			x					x					Mapping macro

Figure B-1 (Part 4 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
ICHPT (IHAICHPT)	x							x					
				x					x				
							x		x		x		
ICSE (IRAICSE)	x										x		
ICSM (IRAICSM)	x										x		
ICSS (IRAICSS)	x										x		
ICT (IRAICT)	x								x		x		
				x					x		x		
IDAL (IECDIDAL)		x							x			x	
IEAPMNIP	x								x			x	
IEAPPNIP	x								x			x	
IEAPQSR	x									x			
IEAPXNIP	x									x			
IEAVNPB	x									x			
IEAVSPSA	x								x			x	
IECDCPT	x									x			
IEEMBBQE		x							x			x	
IEEMBWKA		x							x			x	
IEESMFID		x							x			x	
IEEVC101	x								x			x	
IEEVMENT				x					x		x		User-specified
IEEVMPRM				x					x		x		User-specified
IEFVMSWA		x							x			x	
IEFZB4D2	x								x			x	
IHAGSDA		x							x		x		
IHARCT (IHARCT)			x						x			x	
IHARRRA		x							x			x	
IHASNAP	x								x		x		User-specified
IHSA (IHAIHSA)	x								x			x	
INTWA (IEFVMIWA)		x							x			x	
			x						x			x	
IOCX (IECDIOCX)	x								x			x	
IOE (ILRIOE)	x									x			
IORB (ILRIORB)	x										x		
				x					x		x		
IOSB	x								x		x		The IOS caller determines the location of the IOSB.
		x							x		x		
			x						x		x		
					x				x		x		
IQE (IHAIQE)	x								x			x	
			x						x			x	

Figure B-1 (Part 5 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
IRB (IHARB)	x								x			x	See RB in this table.
ITK (IEFVKEYS)		x							x			x	
IVT (IHAIVT)	x							x			x		
				x					x		x		
JCTX (IEFJCTX)		x							x			x	
			x						x			x	
				x					x			x	
					x				x			x	
JESCT (IEFJESCT)	x								x			x	
		x							x			x	
				x					x			x	
JFCB (IEFJFCBN)		x							x			x	
			x						x			x	
JMR (IEFJMR)		x							x			x	
JSBVT (IEFJSBVT)		x							x			x	
JSCB (IEZJSCB)	x								x			x	
LCCA (IHALCCA)	x								x			x	Fetch-protected
		x							x			x	
				x					x			x	
							x		x			x	
LCH (IECDLCH)	x									x			
LDA (IHALDA)	x								x		x		
			x						x		x		
LLCB (IHALLCB)		x						x			x		
LLPM (IHALLPM)		x						x				x	
LMSG (IECDLMSG)	x								x			x	
LPBT (IRALPBT)	x							x			x		
LPDE (IHALPDE)	x								x			x	
LRB (IHALRB)	x								x		x		
				x					x		x		
					x				x		x		
MCHTRACE	x							x			x		
MCT (IRAMCT)	x								x		x		
				x					x		x		
					x				x		x		
MIHW (IGFDMIHW)	x									x			
MPL (IHAMPL)	x								x		x		
MSG (IGFMSG)	x										x		
MSRASDCA (IEEZB808)	x							x				x	
			x						x			x	
NEL (IEFNEL)		x							x			x	
NVT (IHANVT)	x								x			x	

Figure B-1 (Part 6 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
OLST (IRAOLST)				x				x			x		
ORB (IHAORB)	x							x			x		
OUCB (IRAOUCB)	x								x		x		
				x					x		x		Some fields have new offsets. The Release 2.1.3 OUCB is incompatible with the OUCB in previous releases.
					x				x		x		
							x		x		x		
OUXB (IHAOUXB)	x								x		x		
PCCA (IHAPCCA)	x								x			x	
				x					x			x	
PEL (ISGPEL)		x							x		x		
			x						x		x		
PEXB (ISGPEXB)		x							x			x	
PIPL (IEZPIPL)	x								x			x	User-specified
PQA	x									x			
PQE (IHAPQE)	x									x			Replaced by IHARD
PRULE (IEZPRULE)	x								x			x	User-specified
PSA (IHAPSA)	x								x			x	The upper 2 K is fetch-protected.
		x							x			x	
			x						x			x	
				x					x			x	
					x				x			x	
PTUD (IOSDPTUD)	x										x		
PVT (IHAPVT)	x								x			x	The old PVT is split among IHAPVT, IARRIT, and IARRCE.
		x							x			x	
				x					x			x	
P25C (EWAP25C)		x							x		x		
QCB (ISGQCB)			x						x			x	
QEL (ISGQEL)			x						x			x	
QHT (ISGQHT)		x							x			x	
QRO (IHACTM)		x							x			x	
QVOD (IHAQVOD)				x					x		x		User-specified
QVPL				x					x		x		User-specified
QVPLXVPL				x				x			x		User-specified
QWA (ISGQWA)			x						x			x	
QWB (ISGQWB)			x						x			x	
QXB (ISGQXB)			x						x			x	

Figure B-1 (Part 7 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
RB (IHARB)	x								x			x	
RCE (IARRCE)	x							x			x		
		x							x		x		
				x					x		x		
RCT (IRARCT)	x								x		x		
				x					x		x		
RCTD (IEARCTD)	x								x			x	
RD (IHARD)	x							x			x		
RDCM (IEERDCM)	x								x			x	
			x						x			x	
RESV (IOSDRESV)		x							x			x	
RGR (IHARGR)	x							x			x		
RMCA (IRARMCA)	x								x		x		
				x					x		x		
							x		x		x		
RMCT (IRARMCT)	x								x		x		
		x							x		x		
				x					x		x		
							x		x		x		
RMEP (IRARMEP)	x								x		x		
RMEX (IRARMEX)	x								x		x		
RMPL (IHARMPL)	x								x			x	
RMPT (IRARMPT)	x								x		x		
		x							x		x		
RMQH (IRARMQH)	x										x		
RMSB (IRARMSB)	x								x		x		
							x		x		x		
RNLE (ISGRNLE)			x						x			x	
RPT (ISGRPT)		x							x			x	
RQE (IECDRQE)	x								x			x	
		x							x			x	
RQSV (IRARQSRV)	x										x		
				x					x		x		
RRQ (IECDRRQ)		x							x			x	
RSMHD (IHARSMHD)	x									x			Replaced by IARRAB
RTCT (IHARTCT)	x								x			x	
		x							x			x	
			x						x			x	
RTSD (IHARTSD)	x								x			x	
RVT (IHARVT)	x											x	In the non-page protected nucleus
SCA (IHASCA)	x								x			x	
SCB (IHASCB)	x								x			x	
SCCB (IHASCCB)				x				x			x		
							x		x		x		

Figure B-1 (Part 8 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
SCD (IECDSCD)	x									x			
SCHIB (IHASCHIB)	x							x			x		
SCL (IEEZB815)			x					x					Mapping macro
SCSR (IEZVG100)		x						x				x	
SCT (IEFASCTB)			x						x			x	
					x				x			x	
SCVT (IHASCVT)	x								x			x	
			x						x			x	
SCWA (IHASCWA)	x							x			x		
		x							x		x		
			x						x		x		
					x				x		x		
							x		x		x		
SDEPL (IHASDEPL)	x							x				x	
		x							x			x	
SDUMP (IHASDUMP)	x								x		x		User-specified
SDWA (IHASDWA)	x								x			x	
		x							x			x	
				x					x			x	
					x				x			x	
SIAB (IECDSIAB)	x									x			
SIOT (IEFASIOT)		x							x			x	
SJDFP (IEFSJDFP)		x						x			x		
SJDLP (IEFSJDLP)		x						x			x		
SJEXP (IEFSJEXP)		x						x			x		
SJFNP (IEFSJFNP)		x						x			x		
SJGEP (IEFSJGEP)		x						x			x		
SJINP (IEFSJINP)		x						x			x		
SJJDP (IEFSJJDP)		x						x			x		
SJPRFX (IEFSJPFX)		x						x			x		
SJPUP (IEFSJPUP)		x						x			x		
SJREP (IEFSJREP)		x						x			x		
SJRUP (IEFSJRUP)		x						x			x		
SJWRP (IEFSJWRP)		x						x			x		
SLCA (IFASLCA)		x						x				x	
SMCA (IEESMCA)	x								x			x	
		x							x			x	
				x					x			x	
SMDLR (IHASMDLR)	x								x			x	
	x				x				x			x	
SPCT (IHASPCT)	x									x			Replaced by IARSFTE
SPQA (IHASPQA)	x							x			x		
SPQE (IHASPQE)	x								x		x		
SPT (IHASPT)	x							x			x		

Figure B-1 (Part 9 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
SQAT (IHASQAT)	x							x			x		
SRB (IHASRB)	x								x		x		User-specified
SRCD (ADYSRCD)		x						x				x	
SRIO (IOSDSRIO)			x						x			x	
SRPL (IEEZB814)	x							x				x	
SSAL (IEFSSAL)		x							x			x	
SSCM (IEFSSCM)		x							x			x	
SSCVT (IEFJSCVT)		x							x			x	
SSDA (IEFSSDA)		x							x			x	
SSJS (IEFSSJS)		x							x			x	
				x					x			x	
SSL (IHASSL)	x							x			x		User subpool and key
SSOB (IEFJSSOB)	x								x		x		If the subsystem(s) given control runs in 31-bit addressing mode, the SSOB can be above 16 Mb.
SSOBH (IEFSSOBH)	x								x			x	
SSRB (IHASSRB)	x								x		x		
SSSO (IEFSSSO)				x					x			x	
SSVS (IEFSSVS)	x								x			x	
SSWT (IEFSSWT)		x							x			x	
STCB (IHASTCB)					x			x			x		
STKE (IHASTKE)	x								x			x	
SUB (IEECSUB)	x								x			x	Resides in the nucleus
SUPVT see SVT													
SVCTABLE (IHASVC)	x								x			x	
SVT (IHASVT)	x								x			x	
				x					x			x	
					x				x			x	
							x		x			x	
SWB (IEFSWB)		x						x				x	
SWCT (IRASWCT)				x				x			x		
S99PARMS (IEFZB4D0)				x					x		x		
TAXE (IKJTAXE)	x								x			x	
TBUF (IHATBVT)	x							x			x		
TBVT (IHATBVT)	x							x			x		
			x						x		x		
TCB (IKJTCB)	x								x			x	
				x					x			x	
					x				x			x	
TCCW (IECDTCCW)	x								x			x	
		x							x			x	

Figure B-1 (Part 10 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
TCT (IEFTCT)	x								x			x	
		x							x			x	
			x						x			x	
				x					x			x	
					x				x			x	
TDCM (IEETDCM)	x								x			x	
		x							x			x	
			x						x			x	
TFWA (IHATFWA)			x						x		x		
TOB (IHATOB)	x							x			x		
TOT (IHATOT)	x							x				x	
TPC (IEAVVTPC)			x						x			x	
TQE (IHATQE)	x								x		x		User-specified
			x						x		x		
Trace table entries:													Mapped by IHATTE
ACR	x							x			x		
ALTR	x							x			x		
BRANCH TRACE	x							x			x		
CALL	x								x		x		
CLKC	x								x		x		
DSP	x								x		x		
EMS	x								x		x		
EXT	x								x		x		
IO	x								x		x		
I/O instructions	x							x			x		
MCH	x							x			x		
PC trace	x								x		x		
PGM	x								x		x		
PT trace	x								x		x		
RST	x							x			x		
SIO	x									x			
SRB	x								x		x		
SS	x								x		x		
SSAR trace	x								x		x		
SSRB	x								x		x		
SUSP	x							x			x		
SVC	x								x		x		
SVCR	x								x		x		
SVCE	x								x		x		
USRn	x							x			x		
WAIT	x								x		x		
TRBP (IHATRBPL)	x										x		User-specified
TREP (IHATREPL)	x										x		User-specified
TROB (IHATROB)	x							x			x		
TRQE (IRATRQEL)	x										x		User-specified

Figure B-1 (Part 11 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
TRVT (IHATRVT)	x							x				x	
			x						x			x	
TTCH (IHATTCH)	x							x			x		
TTE (IHATTE)	x							x			x		See trace table entry in this table.
			x						x		x		
TXTFT (IEFTXTFT)		x							x			x	
UCB (IEFUCBOB)	x								x			x	
		x							x			x	
			x						x			x	
UCB look-up table	x									x			
UCDX (IEEUCDX)			x						x			x	
UCM (IEECUCM)	x								x			x	Resides in the nucleus
		x							x			x	
			x						x			x	
USECB (IOSUSECB)	x								x				
UXIR (IHACTM)			x					x				x	
VCB (IHAVCB)	x								x		x		User subpool and key
VFPM (IHAVFPM)	x										x		User subpool and key
VFQE	x								x			x	
VFVT (IHAVFVT)			x						x			x	
VOID (IECDVOID)	x								x			x	
		x							x			x	
			x						x			x	
VRAMAP (IHAVRA)		x							x			x	
VSL (IHAVSL)	x										x		User subpool and key
VSRI (IEEZB812)			x					x				x	
VSSA (IHAVSSA)					x			x			x		
VTPC (IEAVVTPC)	x								x			x	
VTSP (IEFVTSPL)		x							x			x	
WMST (IRAWMST)	x								x		x		
							x		x		x		
WPGDT (IRAWPGDT)	x										x		
			x						x		x		
				x					x		x		
							x		x		x		
WPL (IEZWPL)	x								x		x		User-specified
			x						x		x		
WQE (IHAWQE)			x						x			x	
		x							x			x	

Figure B-1 (Part 12 of 13). Control Block Updates

Control Block	Release							New	Updated	Deleted	Maybe Above 16Mb	Always Below 16Mb	Notes
	2.1.0	2.1.1	2.1.2	2.1.3	2.1.3 VFE	2.1.3 AE	2.1.7						
WSAVTC (IHAWSAVT)	x								x		x		Component-specified, fetch protected
		x							x		x		
			x						x		x		
				x					x		x		
					x				x		x		
WSAVTG (IHAWSAVT)	x								x		x		Component-specified, fetch protected
				x					x		x		
WSAVTL (IHAWSAVT)			x								x		Unchanged
				x					x		x		
					x				x		x		
WWB (IHACTM)		x							x			x	
			x						x			x	
XCPS (IECDXCPS)		x							x			x	
			x						x			x	
XDBA (IECDXDBA)	x								x			x	
		x							x			x	
XFRR (IECDXFRR)		x							x			x	
XPTE (IARXPTE)				x					x		x		
XSA (IEEXSA)	x								x			x	
			x						x			x	
XSB (IHAXSB)	x								x		x		User-specified
XTLST (IHAXTLST)	x								x			x	
XV (IHACTM)		x							x			x	
XVSAV (IHACTM)	x								x			x	

Figure B-1 (Part 13 of 13). Control Block Updates

Index

B

C

D

E

F

G

H

I

J

N

Q

R

T

Numerics

MVS/Extended Architecture
Conversion Notebook
Volume 1

READER'S
COMMENT
FORM

GC28-1143-6

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

MVS/Extended Architecture Conversion Notebook Volume 1

GC28-1143-6 S370-34

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®

GC28-1143-06

MVS/Extended Architecture Conversion Notebook Volume 1

GC28-1143-6

S370-34

Printed in U.S.A.